INFORMATION SYSTEMS DEVELOPMENT THROUGH AN INTEGRATED FRAMEWORK

MOHAMMED HASAN SALAHAT

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the degree of Doctor of Philosophy

School of Computing and Engineering

University of Huddersfield

Huddersfield, United Kingdom

December, 2016

PUBLICATIONS

1-Salahat, M. and S. Wade (2016). "Domain Driven Design vs Soft Domain Driven Design Frameworks". International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol: 10, No: 7, 2016.

2-Salahat, M. and S. Wade (2015). "Business Domain Modelling Using an Integrated Framework". International Journal of Social, Behavioural, Educational, Economic, Business and Industrial Engineering Vol: 9, No:6, 2015.

3-Salahat, M. and S. Wade (2014). "Teaching Information Systems Development through an Integrated Framework. UKAIS2014 Conference, Oxford University, Oxford.

4-Salahat, M. and S. Wade (2012). Pedagogical Evaluation of a Systemic Soft Domain-Driven Design Framework. The UKAIS International Conference on Information Systems 2012 Proceedings, AIS.

5-Wade, S., et al. (2012). "A Scaffolded Approach to Teaching Information Systems Design." Innovation in Teaching and Learning in Information and Computer Sciences 11(1): 56-70.

6-Salahat, M. and S. Wade (2009). A Systems Thinking Approach to Domain-Driven Design. UK Academy for Information Systems Conference Proceedings 2009.

7-Salahat, M., et al. (2009). "The Application of A systemic Soft Domain Driven Design Framework". International Science Index, International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:3, No:9, 2009

8-Salahat, M., et al. (2008). A systemic framework for business process modelling and implementation. Innovations in Information Technology, 2008. IIT 2008. International Conference on, IEEE.

9-PhD Consortium presentation in UKAIS09, Oxford University, Oxford, UK, 2009.

10-Salahat, M. (2008) A systemic framework combing soft and hard systems development techniques for BPM and Implementation. Poster displayed in the Annual Researchers' Conference 2008, University of Huddersfield, UK.

Abstract

Information systems are essential entities for several organizations who strive to successfully run their business operations. One of the major problems faced by the organizations is that many of these information systems fail, and thus the organizations do not achieve their required targets in time. Many of the reasons for the information system failures documented in the literature are related to development methodologies or frameworks that are unable to handle both 'hard' and 'soft' system aspects. In general, the hard issues of the system are considered more significant than the soft issues, however, all the methodologies must be able to deal with all the system and business aspects.

This thesis investigates the possibility of developing and evaluating a multimethodology framework that can be used for information systems development in an academic and business environment. The research explores the applicability of such a framework that comprehends both 'soft' and 'hard' system aspects in order to eliminate information system failures. Different software development approaches are investigated, including the dominant 'domain-driven design' (DDD) approach.

A new multimethodological framework entitled 'Systemic Soft Domain Driven Design' (SSDDDF) has been developed by combining 'soft system methodology' as a guiding methodology, 'unified modelling language' as a business domain modelling approach, and a domain-driven design implementation pattern. This framework is intended as an improvement of the DDD approach. Soft and hard techniques are integrated through mapping from the 'consensus primary task model' of the soft approach to the 'use cases' of the hard approach. In addition, 'soft language' is introduced as a complement to DDD's 'ubiquitous language', for facilitating the communication between the different stakeholders of a project. The implementation pattern (e.g., Naked Objects) is included for generating code from domain models.

The framework has been evaluated as an information systems development approach through different undergraduate and postgraduate projects. Feedback from the developers has been positive and encouraging for further improvements in the future. The SSDDD framework has also been compared to different ISD methodologies and frameworks among of these DDD as an approach to ISD. The results of this comparison show that SSDDDF has advantages over DDD and significant improvements to DDD have been achieved.

Finally, the research suggests an agenda for further improvements of the framework, while suggesting the development of different pattern languages.

Table of Contents

List of Tables

List of Figures

List of Appendices

Acknowledgements

Special thanks are due to all the people who have encouraged me to complete this research work, especially:

To my beloved wife Dalal - you are an amazing gift, an exceptionally excellent woman and the best wife. Thank you for supporting me to finish this work.

To my little cute daughter Layan, who always prays to God to help 'Baba' get the doctorate and come back home safely.

To my other daughters and sons, Nariman, Doaa, Rawan, Razan, Ahmed, Saja, and Mustafa - thank you for your patient acceptance of me leaving you alone for a long period.

To my father, who died while I was in the early stages of this work.

To my mother - I hope for her good health.

Special thanks to my kind supervisor Dr. Steve Wade for his cooperation and support.

Special thanks also to my co-supervisor Professor Jun Lue, and to the school research office team, especially Gwen and Chris.

To all friends and colleagues in Ajman University, UAE, who have always encouraged me to complete this work.

Chapter 1: Introduction

1.1 Background and Motivation

Information system (IS) is defined by (Davis, 2000) as an organisational system that delivers information and communication services required by the organization. A comprehensive definition of IS comes from Laudon & Laudon (2009) where they defined the IS as "related parts working together to collect, process, store, and produce information for supporting decision making, coordination, control, analysis, and visualization in an organization". Zwass (2016), defined IS as "integrated set of components for collecting, storing, and processing data for providing information, knowledge, and digital products" . The literature on IS has emphasized on its application among the computer-based information and communication tools and the difficulties in understanding and developing information systems for effective utilization.

IS addresses several issues that might improve the organisational operations, for instance, facilitating organisational every-day operations, simplifying the interaction process between the organisation, customers and suppliers, and improving the organizational performance and profitability (Devaraj and Kohli 2003; Hendricks et al. 2007; Melville et al. 2004; Sabherwal et al. 2006). Therefore, IS might add a competitive edge for the organisation in the marketplace (Zwass, 2016). With the progressive development in technology, organisations utilise IS to facilitate the execution of different tasks with accuracy and preciseness. Also, time is one of the key factor that assists in improving the organisation's work and performance. IS, performs complex tasks with minimum intervention from the users, and hence, consumes less time while increasing the efficiency ((T Bhuvaneswari & S Prabahara, 2013).

Globalization and high competitive environment has compelled the organizations in improving their information systems for meeting the demands of the emerging markets (Kaur & Aggarwal, 2013). However, several critical issues are encountered in an information system that must be handled in order to ensure the achievement of the desired goals. IS failure, where information systems are unable to meet the user expectations, create a working or a functioning system (Ewusi-Menash 2003), encounter a budget overrun, have a

late delivery, and fail to achieve objectives are the impending issues. Information system and its management experiences high failure rate, either total or partial. IS failure can have more severe consequences where the system stops running completely (total), or some of the system functions do not working properly (partial). Also, the failure can be temporarily (a day or few days) due to some technical and non-technical problems (Donaldson, A. J. M., & Jenkins, J. O., 2001). Hence, the organizations do not achieve their required targets in using IS, and IS failures might cause financial loses.

Different reasons might contribute in IS failures. For instance, the organizational structure and culture factors are found to cause IS failures (Lavallée, M., & Robillard, P. N., 2015). Language and cultural barriers among the IT developer and user can create disappointment in the developed IS and cause a complete failure. The other reasons of information system development failures are inadequate support/leadership from senior management, ignorance towards the stability of the technology used, lack of efficient communication and failure to manage complexity (Kaur & Aggarwal, 2013). Lack of cooperation within the teams, lack of standardization, lack of devotion, no availability of data and lack of management support are some of the other factors that affect the successful development and implementation of information systems (Al-Mahid & Abu-Taieh, 2006). In addition, wrong choice of Information System Development Methodology (ISDM) or framework are also potential reasons for failure information systems (Charvat, 2003; Sauser et al., 2009). There are different definitions of ISD, and the most comprehensive definitions are:

(Avison and Fitzgerald, 1995) provided comprehensive definition: "a collection of procedures, techniques, tools, and documentation aids which will help the systems developers in their efforts to implement a new information system. A methodology consist of phases, themselves consisting of sub-phases, which will guide the systems developers in their choice of the techniques that might be appropriate at each stage of the project and also help them plan, manage, control and evaluate IS projects",.

(British Computer Society, 2006) defined ISDM as "A recommended collection of philosophies, phases, procedures, rules, techniques, tools, documentation, management, and training for IS developers",.

The purposes of using ISDM(s) are to investigate and gather the system requirements in order to develop information system to support the organizational needs. ISDM might

capture all information needed from the business domain, and this information should be used throughout the IS development process.

Information systems are distinguished from other fields on the basis of its foundation pertaining to the "artefacts in human machine systems", where the focus in laid on the human elements in an organizational system. Thus, the information system refers to both the aspects, soft and hard (Hasan, 2003). The ISDM that are unable to handle the information systems perspectives (both 'soft': "human-centred" and 'hard': "technology-centred") causes the IS failure. Several information systems have failed, which are usually attributed to poor business process modelling (Barjis, 2008). The design and implementation of information systems within an organization have found to cause challenges and failures due to their incompatibility with the business process models. It has been argued that one of the major reasons for information systems failure is the tendency to concentrate more on the technical aspects (hard aspects) of the design rather than acquiring a thorough understanding of the business needs (soft aspects), thus, leading to a poor business process model which might not adequately support the design and implementation of the IS (Alter, 2006).

It is also argued that the adapted methodology or framework might use the business modelling to create an abstraction of the business in order to get a clearer understanding of its information requirements, so as to improve the current process (Alzubidi, Recker & Bernhard, 2011). There have been a number of attempts to develop business models using hard approaches, such as the unified modelling language (UML) which is primarily an object-oriented modelling approach that can model the hard aspects of business processes in different diagrams. Using the same modelling language to represent the business and the software that supports it is attractive. If this is possible, clearer communication can be expected between people who are involved in managing the business and those responsible for developing the IS. UML can be used for the analysis and design of system processes to acknowledge the business needs before the development of the information systems (Yusuf et al., 2011). However, UML cannot handle soft issues, and only considers the 'hard' system requirements. Therefore, soft system methodology (SSM) is used for information system analysis to deal with the soft issues. SSM is an approach to business process modelling that can be used for both general problem solving and management of change. The approach has been most successful in the analysis of complex situations because there are different views about the problem identification and definition(i.e. 'soft problems').

Understanding the business needs and inculcating them in the development of information systems contributes to the successful compilation of the system without any failure. Also, determining and understanding 'soft' and 'hard' business systems aspects is highly important for developing information systems which are expected to reflect business needs. Therefore, in order to consider both the soft and hard issues, a combination between UML and soft approaches like soft system methodology are encouraged (Checkland, 1981; Bustard et. al, 1999; Sewchurran & Petkov, 2007; Al Humaidan, 2006).

However, as mentioned above, the software engineering and the development approaches of information systems are rich in complexity and beset with challenges, resulting in IS failures. The development of information system is integrally complex as it addresses both technological challenges and organizational issues that falls out of the project scope. Also, the organizations strive to use their existing systems and integrate changes within them with new development efforts that further increases the complexity. Further, the dynamic business requirements and organizational needs have created difficulties in developing a system that fulfils all the requirements and system specifications (Xia & Lee, 2005). Handling the complexity of IS development projects have been the most essential responsibilities of IS managers in an organization, hence, presenting a dire need of investigation. Therefore, to reduce the complexity of processes in an information system development, it is crucial to follow a systematic approach of development. This indicates that a systematic approach (framework) is required for capturing the information from business processes(business domain), and to explore their models in an aproper way to enable the IS development and avoid IS failures (Sewchurran & Petkov, 2007; Al Humaidan, 2006; Strong and Volkoff 2010; Volkoff et al 2007). Also, this addresses the need to bridge the gap between 'business process modelling and implementation', in order to model and implement the business domain model as IS.

Through this holistic view of IS failure, this thesis attributed the IS failure reason which belong to "wrong choice of the ISDM or framework used to develop IS". The thesis aims to investigate ISDM and explore the possibility of developing and evaluating a multimethodology framework for information systems development, and its applicability to a consideration of both 'soft' and 'hard' system aspects which might help to eliminate the IS failures.

1.2 Business Domain Modelling and Implementation

A business domain model consists of structural and behavioural components. The structural part provides an understanding of business artefacts and determines the relationship between them, while the behavioural part corresponds to the business processes of the business domain (Bennett, 2007).

The business domain models adopted by different organizations are similar but differ in terms of perspectives (Oldfield, P., 2002). Domain model represents the application domain that facilitates communications between business experts and IT through ISDM (Rose J., 2002). Therefore, the challenge is to adopt a framework which provides the project team with the required tools for modelling the business artefacts and also allows an easier mode of interaction. If an appropriate framework is adopted, then the organizations can build a proper business domain model which can be mapped into IS during later stages. Based on this knowledge, it is argued that there is a need to adapt an understandable language for the team members to interact between business domain investigation phases until the code generation phase. The ICONIX process (is a use case driven process and it's consist of four millstones for Information Systems Development (ISD), (Rosenberg & Stephens, 2007) supported this idea and concentrated on the importance of having a common communication language to facilitate communication between the team throughout all phases of a project. Domain-driven design, or DDD, (Evan, 2004), is a software development approach which adopts a 'ubiquitous language' as a communication language between the project team. This language is the backbone of the model and the base for the developers and the business experts to have a common understandable communications between them through the development of IS phases.

1.2.1 Domain-Driven Design

Domain-Driven Design (DDD) models business processes as a 'domain model' that can be mapped automatically into object-oriented codes to produce an information system (Evan, 2004). This approach concentrates on a clear understanding of the business domain by utilizing a 'ubiquitous language' as a communication tool between different stakeholders (business experts and developers). The other mechanisms utilized by DDD approach is UML modelling and object-oriented programming languages. The basic idea of DDD is that the design of the software must reflect the business domain in order to develop the requested information system. DDD assumes that the business experts will become familiar with the related diagrams and tools through the discussion, but because these techniques are usually

mastered by the developers and not business experts, the idea of 'knowledge crunching' (Evan, 2004) is used, which consumes more time in understanding the technical aspects of the language. However, it may be argued that business experts will encounter difficulties in 'crunching the knowledge' and understanding these tools. Therefore, there is a need to reconsider and modify the structure of the language to make it more comprehensive for different stakeholders, especially business experts. This is considered as a potential gap of this approach, which requires improvement of the 'ubiquitous language' into new version called 'soft language', as proposed in this thesis.

1.2.2 Hard Approaches

ISDM can be grouped into soft and hard methodologies. One classification approach has been classified hard methodologies into traditional approaches (heavyweight) and Agile approaches (lightweight) (Boehm & turner, 2003; Charvat, 2003; Highsmith, 2013; Wysocki, 2009). Heavy weight like Waterfall (Benington, Herbert D., 1956, 1987), Iterative Waterfall (Winston Royce, 1970), Waterfall (Bell, Thomas E., and T. A. Thayer. , 1976), B-Model (Birell and Ould, 1985), Information engineering (Martin & Finkelstein, 1981), Spiral model (Barry Boehm, 1988), Structured Systems Analysis and Design Methods (SSADM) (Ashworth and Goodland,1990), Unified Software Development Process (Jacoboson, Booch, & Rumbaugh, 1999), prototype model (Pressman R. S., 1994), and Microsoft Solution Framework (MSF) model (Microsoft, 2004). Other classification approach classified hard approaches into models based on sequential approach like waterfall model, and models based on iterative approach like prototype model, spiral model, unified process model, Microsoft solution framework, and agile methods (Predrag Matkovic & Pere Tumbas, 2010). Other approach classified hard approaches into structured methodology and object oriented methodology.

Agile methods are not fixed and standard steps, but are base methods that can be modified from one project to another. Agile approaches require a base method to be configured by comparing the conceptual model of the information system development process with the requirements of the project being developed. These methods aimed to provide sufficient processes for any given project but tried to avoid detailed descriptions of processes (Ambler, 2002). Later, object programming languages such as Java and C# were introduced, and these languages were supported by object-oriented analysis and design and o-o relational databases. These languages facilitated the agile methods by increasing the speed of developers in programming, without wasting time in the design details.

The unified modelling language (UML) was introduced by Fowler and Scott (2000) as a means of representing object-oriented programming design. Later, this became a standard for software design. UML consists of a group of diagrams to describe the software system. Different development methods have adopted UML diagrams, such as the 'Unified Software Development Process' (USDP) (Jacobson, I., Booch, G. and Rumbaugh, J., 1999) and the 'Rational Unified Process' (RUP) (Kruchten, 2004; Manalil, J. (2011)). Some of the other agile methods also became familiar in use, such as Alistair Cockburn's 'Crystal' family of methods (Cockburn, 2001), Peter Coad's 'Feature Driven Development' (Coad, 1999) and Jim Highsmith's 'Adaptive Software Development' methods (Highsmith, 20013). Agile methods are base methods that can be modified from one project to another. The base method is configured by comparing the conceptual model of the software development process with the requirements of the project being developed.

Systems requirements consist of hard and soft aspects. The hard systems approach deploys methods for designing an optimal solution for the development of information systems, it however lacks in terms of comprehending the 'human' element. All hard approaches focus on the systems' and users' requirements, which are mainly classified under 'system hard aspects'. It keeps the technical aspects on priority and follows a scientific approach to problem-solving. However, soft aspects are also important parts of any system and must be considered. Therefore, soft system approaches were developed in 1980s to incorporate the human element in the development of information systems (Van de Kar & Verbraeck, 2008). Soft system methodology is one of the most extensively used approach in soft systems, which is briefly described in the next section.

1.2.3 Soft Systems Methodology

Checkland, 1981 and other researchers developed a methodology called Soft systems methodology (SSM) at Lancaster University. It is a problem-solving methodology which focuses on the soft issues of a system and is applied to investigate problematic situations (Checkland & Scholes, 1990; Checkland & Howell, 1997). Soft system approaches or SSM assumes that human factors are highly essential, the stakeholders consider the problems differently, and the outcomes must be learning and better than solutions. SSM focuses on the development of conceptual models of the system which will be compared to the existing real world model. This approach can be used to investigate different systems in different situations, and it contributes to the analysis of information systems design. SSM is a methodology which is well known for dealing with soft system aspects, exploring problem

situations and modelling human activities using different diagrams such as rich picture and human activity system diagrams. These diagrams are not technical but seek to represent the real world as an abstract model. However, the high complexity and difficult management of business projects concerned with information system, requires a more efficient approach to tend to both the soft and hard issues, as SSM gives priority to soft systems. Therefore, integrated approaches have been developed to incorporate both these aspects.

1.2.4 Soft and hard aspects in software design and development

Software development has been rendered as a domain that addresses socio-technical aspects, where the focus has been laid on the need to communicate between the users and developers (Ahmed et al., 2013). The development of software and information system is reliant on two aspects, soft, which refers to the problem solving capabilities, social interaction and human needs, and hard, which refers to the technical perspective of developing a system. While selecting an information system methodology to solve a problem, a distinction between hard and soft problems must be considered to guide the selection. Hard problems emphasise on answering the question of 'how' a system has to be developed. With hard problems, there is a solution by which the aims are achieved. Hard approaches to system development have been succeeded in developing information systems from the technical perspectives. On the other hand, as mentioned by Curtis (1992), the information system sometimes rejected by the user as they are unable to comprehend its utility. This raise the issue that an alternative approach is required to capture the human elements (soft) of a system. Therefore, it is essential to incorporate both the hard and soft system requirements to fulfil the success of various applications where information systems are developed. The different approaches to utilize these hard and soft aspects have been further explored in the literature review chapter of this thesis.

1.2.5 Combining SSM and UML

There are different researchers trying to integrate SSM with structured development methods (Keys, P. and Roberts, M., 1991; Lewis, P., 1995; Miles, R., 1992; Mingers, J., 1988; Prior, R., 1990. More recently, some efforts to integrate SSM with object oriented were made, (Bustrad, et al, 2000) which executed the integration of SSM with UML use cases. The work in this area demonstrates the importance of such integration for investigating a complex and messy problem situation. Other research efforts clarified that using techniques from hard approaches alone (e.g UML) is not applicable when the

requirements unclear and the combination between SSM and UML is required to evaluate the requirements from the perspectives of different stakeholders (Bustrad, et al, 1999; Steve W. & Judith Hopkins, 2002; Al Humaidan, 2006; Sewchurran & Petkov, 2007). They concluded that the combination of SSM and UML encouraged the SSM exploration of system activities from the system itself and their conversion into use cases (representing the system activities) from the users' perspectives (Bustrad, et al, 1999), and combining UML with SSM might help in modelling both 'hard' and 'soft' system aspects of the business domain to develop IS, which are expected to reflect business needs (Al Humaidan, 2006; Sewchurran & Petkov, 2007; Bustrad, et al, 1999; Steve W. & Judith Hopkins, 2002). This combination is achieved using use case diagrams that will accommodate all the knowledge generated by SSM conceptual models during the business domain investigation phase. The combination of SSM and UML is expected to provide a good improvement to the modelling and implementation of businesses processes within the business domain, and to contribute to the elimination of information systems failure.

Recent work shows that the combination between SSM and UML is used to contextualise the problem space using SSM and developing UML models to solve the complex problems (Ross Fenning et al, 2014) to design a complicated search engine for BBC (British Broadcasting Corporation). Other recent works have presented systems thinking-based approach for finding the requirement in complex situations, by exploring and identifying the challenges of complex situation requirements gathering to be the requirements nature, the observer role, and the system environment (Polinpapilinho F. Katina, Charles B. Keating, Ra'ed M. Jaradat, 2014). Minger (2001) added that gathering understandable, consistent, modifiable, and verifiable requirements is difficult with the complex situation. Further, to achieve such requirements, a change in paradigm is required such as an integrated multiple infrastructure through holistic thinking, as done in this thesis to mix different methods from different paradigms to deal with complex situation (Minger, 2001).

The combination of SSM and UML is expected to provide a good improvement to the modelling and implementation of businesses processes within the business domain, and to contribute to the elimination of information systems failure. It is further argued that using SSM to explore the business domain may be a good addition to the DDD, as SSM can be used at the beginning to explore the problem situation, and both domain experts and developers should share the exploration of the problem and the development of the SSM conceptual models. This may increase the developers' understanding and awareness of the targeted domain, and may help the domain experts in mastering the conceptualizing skills,

which will facilitate their understanding in the later stages of technical modelling. The output of SSM is expected to be a good addition to the ubiquitous language, since it consists of human activity models that can be understood by both business experts and technical staff.

These related works have recognized the need for more investigation of business domain, with more emphasis on soft and hard system aspects that can affect the successful implementation of the information system. This has encouraged the current researcher to use this combinations into a proposed framework that might model and implement the system perspectives of the business domain into an IS which might help to eliminate IS failures. It has further motivated the research in suggesting a complementary language called 'soft language', for the new proposed framework, which is called 'Soft Domain-Driven Design'.

1.3 Research Aims and Objectives

The current research aims at integrating both hard and soft approaches to improve the development of information systems. Also, the study aims to develop a framework that can be effectively deployed in the information system development projects undertaken by students. This thesis addresses several different important issues. First, it describes the problem that most of the multimethodology frameworks are unable to consider. Both, soft and hard systems aspects are considered in exploring and modelling business domain processes. Secondly, it investigates, analyse and models the business domain processes, creating a domain model that reflects the internal business processes of the business domain concerned. The model is then used to implement the target domain into a software system. Thirdly, it focuses on the integration of software development approaches in order to formulate a multimethodological framework that can consider all soft and hard system aspects in the context of business domain process modelling. It demonstrates and use a technique to move from the SSM conceptual model into UML use cases. Finally, it uses the multimethodology as a guided framework for information systems developers to help them through the system development stages step by step.

The proposed framework SSDDDF is based on a multimethodology approach, which justifies the combination of methods for the same business intervention (Mingers, 2001). It is a multi-method framework which is intended to guide developers through the investigation of a problematic situation. Therefore, the purpose of the framework is to achieve a

comprehensive understanding of the systems being developed, and to easily guide developers step by step through what they are developing.

Therefore, the research questions of the research undertaken in this thesis are:

Q1: How can we formulate a multimethodology framework that will allow us to investigate, analyse, model, and implement the business processes from a specific domain by considering all the relevant "soft" as well as "hard" system requirements?

Q2: What benefits can we demonstrate from applying the proposed framework in a number of ISD projects?

1.4 Contributions

As stated above, the development and evaluation of the SSDDD framework has aimed to answer two research questions in order to fill the mentioned gaps in the knowledge. This process has enabled certain contributions to be made by this research, which are outlined as follows:

1- The research proposes and demonstrates the application of a multimethodological framework for information systems development called 'Systemic Soft Domain-Driven Design Framework (SSDDDF) to deal with both 'soft' and 'hard' business domain perspectives as an improvement of DDD. The proposed approach is an improvement to the existing approaches and forms an effective mechanism of comprehending all the requirements of the system.

2- The research introduces a 'soft language', as a complement to DDD's 'ubiquitous language', which consists of SSM modelling tools to facilitate the communications between the ISD project stakeholders. With this language, the communication is increased with high level of clarity.

3- The research demonstrates a technique to perform transition from SSM CPTM (Conscious Primary Task Model, Brian Wilson's, 1990) to UML use case diagram. This technique is demonstrated through different applications of the framework in school projects.

4- The research models the business domain as a 'domain model' (UML Class diagram and other UML supported diagrams), which can be moved directly into

software code through implementation patterns. It further recommends the tools of implementing patterns (Naked Objects or TrueView).

5- Demonstration and practising of how SSDDDF can be used as an ISD framework through different projects taken by students.

1.5 Thesis Outline

This thesis explores how soft systems methodology (SSM) and unified modelling language (UML), as tools of domain-driven design (DDD), can be integrated within a wider framework to increase the effectiveness of requirements modelling for information systems (IS) development. The proposed framework leads to a detailed domain model that is a literal representation of an information system that could be implemented by following the Naked Objects or TrueView (implementation patterns). Within the proposed framework, requirements analysis is conceived as a two-stage process. Firstly, a business analysis is carried out to make sense of the human activities performed in an organization. In this stage, SSM is employed to help users understand what information they need and why (introducing 'soft language' as a compliment of 'ubiquitous language' developed by Eric Evan, 2004). Secondly, a technology-oriented analysis is carried out to define what technological facilities might support the organizational activities. Here, DDD tools (UML and others) help to build a data structure capable of satisfying the information needs identified. The results of DDD are then implemented using the Naked Objects framework or TrueView (implementation patterns). The outline of this thesis is as follows:

Chapter 1: This chapter discusses the background of the study, and explores DDD, business process modelling, soft and hard of information system development, integrating SSM and UML, research aims and objectives and contribution of the study

Chapter 2: This chapter refers to the literature review, which is divided into two parts. Part1 reviews and discusses related works, which include those methodologies and frameworks related to ISDM, business process modelling, and similar multimethodology frameworks in the literature. Part2 provides the descriptions of the selected tools used by the proposed framework like domain-driven design, soft system methodology, and UML, and implementation pattern.

Chapter 3: This chapter presents and describe the methodology adopted in this research to propose and evaluate the framework SSDDD.

Chapter 4: This chapter describes, explains, and illustrates the proposed SSDDD framework by explaining all development process in detail.

Chapter 5: The chapter presents the application and evaluation of the framework by using different projects at undergraduate and postgraduate level.

Chapter 6: Here, the evaluation of the framework is presented through teaching ISD module 'Methods and Modelling' and by comparing it with the domain-driven design and other ISD frameworks reviewed in Chapter 2.

Chapter 7: This is a summary and conclusion chapter which considers the results of all the evaluations presented in other chapters, presents the contribution of this thesis, provides a discussion of the results, and offers recommendations for future investigation.

1.6 Conceptualization of the Thesis

The following Figure 1-1 presents the flow of this thesis through the different chapters and shows how each chapter can be visited through the reading process.

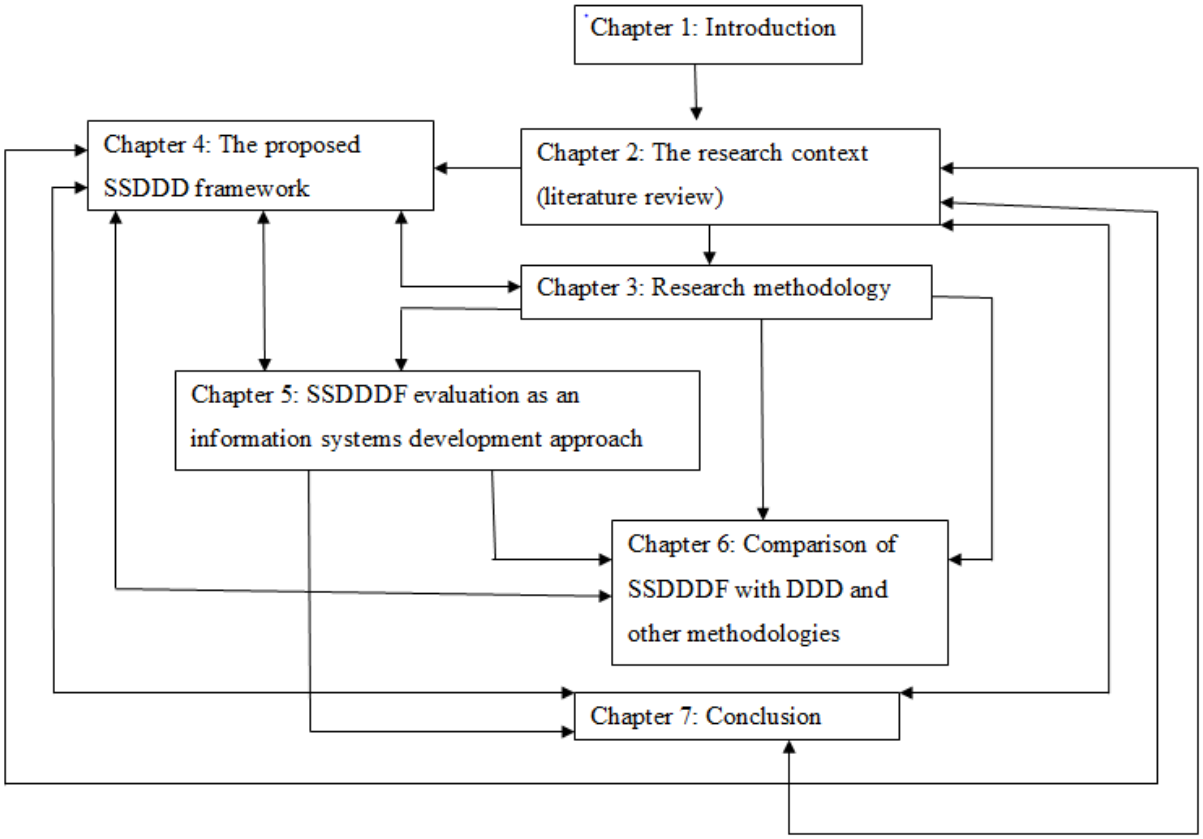


Figure1- 1: Conceptualization of the thesis

Chapter 2: The Research Context (Literature Review and selected ISD Tools)

Part 1: Literature Review

2.1 Introduction

This review critically collecting and evaluating information from the relevant existing literature on information system development and the failures encountered in them. It further explores the business process frameworks, business domain modelling and information system development methodologies. Also, the soft and hard aspects of the IS are explored, wherein the integration of SSM and UML, transition from SSM to UML use cases, domain driven designs and multi-methodology frameworks are discussed. The purpose of this critical literature review is to find and review the available studies related to the research aims and objectives of this research work (explore different contexts and their related research results) and to come with a related conclusions to support this research.

2.2 Introduction to Information Systems

Information system is defined by (Zwass, 2016) as "integrated set of components for collecting, storing, and processing data for providing information, knowledge, and digital products". Davis (2000) defined IS as an organisational system to deliver information and communication services required by the organization. Laudon & Laudon (2009) provided comprehensive definition of IS, where they defined the IS as "Interrelated components working together to collect, process, store, and disseminate information to support decision making, coordination, control, analysis, and visualization in an organization". According to Hasan (2003), information systems (IS) are regarded as the essential attributes in the modern technology that has enabled the intricate combination of socio-technical aspects constituting of hardware, software, people and work processes. An information system comprises of shared technology resources, which are essential for managing the specific information system applications. The other components of IS includes the application software services, telecommunications, resource planning, knowledge management systems and customer relationship management, which are all facilitating the growth of an organization (Kaur & Aggarwal, 2013). For effective business operations, organizations strive to develop and adopt information system tools for enhancing efficiency and

productivity. Information systems processes the input data of an organization and generates valuable information that enables the successful compilation of operations.

Information systems assist the organizations in conducting thorough research, developing and deploying new approaches of conducting business operations, for the purpose of increasing efficiency. Organizations and enterprises have to manage a gamut of data and information from several sources and exploit them to perform business functions, which can be effectively organized by the information systems. These systems are capable of generating automatic steps of performing operations that were once done manually, thus not only increasing accuracy but also saving time. Information systems supports the organization in managing the information, taking critical decisions and implementing the business processes in an efficient way as possible (Laudon, 2009). Advances have been made in information systems with the augmentation of globalizations, where new tools and techniques have not only assisted in saving time in executing business activities, but also reduced the costs of operating and transacting. In recent times, every organization has been equipped with an IT department comprising of IT professionals, managers or outsourced IT services, thus forming the integral component of an organization's infrastructure. The IT department is accountable for managing the hardware, software, and other essential IT services, i.e. developing IS. The information systems are developed in an organization to find patterns in the information and create knowledge for increasing the productivity of the businesses through better decision making via information system intelligence (Laudon, 2009).

2.2.1 Information System Failures

Information systems failure is widely documented in the literature and a variety of different reasons are given for it. According to Kivuva T. (2012), information systems have evolved with time to address organizational needs by not just performing simple computational operations but also acknowledging strategic needs of processes. The researcher investigated the failures and challenges that are encountered in the development and implementation of information systems, and found that scheduling overruns, poor management, organizational politics, slow adaptation of changes and procurement process, poor understanding of requirements, poor IT infrastructure and lack of technical staff are the causes of failures. Therefore, it is inferred that understanding the requirements of the project is essential in the development of the information systems, which must be addressed with changing time,

as with time the requirements may also change. The failures have a significant impact on the efficiency of the operational processes and lead to poor performance of an organization.

According to Lucky & Adegoke (2014), the challenges faced in the development of information systems correspond to the infrastructures (both hardware and software), materials, processes and manpower, and lack of funds which must be addressed for gaining effectiveness. The researchers have further determined that developing a complex information system requires a multimethological approach that is rendered as the most effective strategy. Qualitative analysis were performed by the researchers to reach the conclusions, however, the challenges pertaining to the infrastructure were not well investigated. The study concluded that a well-structured information system is required with a central database to address the challenges and mitigate the causes of failure.

Al-Mahid & Abu-Taieh (2006) discussed the factors that interfere in the successful development of the information system in developing countries. It was revealed that the factors such as the attitudes of developers, poor coordination, lack of data appreciation, computer illiteracy, lack of supporting regulations, lack of collaboration and understanding of requirements causes IS failures. The researchers have emphasized on providing appropriate education and promoting IT to overcome the challenges, however, they have failed to address the challenge of poor understanding of the requirements. Kaur & Aggarwal (2013) have stated that the inability to manage complexity of the information systems is a critical reason of failure that must be acknowledged and resolved. If the challenges are not addressed, then the failure of information systems will have a direct impact on the overall productivity of the organization. Therefore, it is imperative to avoid such failures for the purposes of gaining business advantage in the competitive environment.

Ewusi-Menash (2003) discussed the cases of IS failures where information systems are unable to meet the user expectations, create a working or a functioning system, encounter a budget overrun, have a late delivery, and fail to achieve objectives are the impending issues.

According to Donaldson, A. J. M., & Jenkins, J. O. (2001), Information system and its management experiences high failure rate, either total or partial. IS failure can have more severe consequences where the system stops running completely (total), or some of the system functions do not working properly (partial). Also, the failure can be temporarily (a day or few days) due to some technical and non-technical problems (Hence, the

organizations do not achieve their required targets in using IS, and IS failures might cause financial loses.

Lavallée, M., & Robillard, P. N.(2015) try to be more comprehensive by trying to find different reasons might contribute in IS failures. According to them, the organizational structure and culture factors are found to cause IS failures. Language and cultural barriers among the IT developer and user can create disappointment in the developed IS and cause a complete failure.

Kaur & Aggarwal, (2013) determined other reasons of information system development failures to be inadequate support/leadership from senior management, ignorance towards the stability of the technology used, lack of efficient communication and failure to manage complexity .Lack of cooperation within the teams, lack of standardization, lack of devotion, no availability of data and lack of management support are some of the other factors that affect the successful development and implementation of information systems according to (Al-Mahid & Abu-Taieh, 2006).

Charvat, 2003 and Sauser et al., (2009) illustrated that the wrong choice of Information System Development Methodology (ISDM) or framework are also potential reasons for information systems failure.

Another reason of IS failure include poor business process modelling, concentrating on the technical aspects of design rather than on understanding business needs. The information system refers to both hard and soft aspects, and thus, both of these must be incorporated in the development and implementation of information systems to avoid any failure. This issue is related to the information system development methodology (ISDM), and a number of different methodologies and frameworks are available for developing information systems, some of which have been recently developed. ISDM is the backbone of information systems as it is used to structure, plan and manage the complete procedure of development. It is expected that these frameworks, if applied and used well, will reduce software systems failure, as it will understand the requirements and needs of business operations and inculcate them into the development of the information system. On the other hand, the poor selection of ISDM leads to ISD failure. Therefore, a framework that is able to handle both hard and soft system issues must be developed and adopted (Sewchurran & Petkov, 2007; Al Humaidan, 2006; Strong and Volkoff 2010; Volkoff et al 2007; Bustrad, et al, 1999; Steve W. & Judith Hopkins, 2002). The thesis acknowledged all

reasons of information systems failure as mentioned above in this section. One major reason of information system failure which is related to ISD methodology attributed as an important reason. This lead to the main aim of this research work is to propose a framework which can be used for information systems development. Information systems are used to support an organization and because of this, business process investigation and modelling must take place first, in order to understand what is required from the information system; then, the tools and methods required to handle such a system can be determined.

2.3 Business Processes

The studies of business process dates back to as early as 1911, when Fredrick Taylor researched the effectiveness and efficiency of work procedures in order to improve them. More efforts to improve business processes continued, and ranged from studies in 'business process reengineering' (BPR) by Davenport and Short (2003) and Hammer (2009), to explorations of 'business process management' (BPM) by Goyal, D. P. (2012), with the goal of improving business performance. The transition to BPM occurred because several BPR projects had failed and attracted certain criticisms. More recently, researchers have been concentrating on 'business domain modelling', which aims to distil knowledge from business domain experts in order to create a business domain model and thus develop the software system. One of these researchers is Eric Evan (2004), who focuses on generating business domain models from business domain experts and has introduced the 'ubiquitous language' as part of domain-driven design approach. DDD is considered by this thesis as the main framework for information system development and will be discussed further in part2, section 2.4.

There are different definitions of a business process, which are based on the idea of a deterministic system that receives inputs and transforms them into outputs following a step-by-step series of activities. This perspective is rooted in the idea of production processes, which can be described as a step-by-step procedure of taking raw materials and transforming them into a finished product (Lindsay, A., D., & Lunn, K. ,2003). This machine-model concept of a process has been applied in many fields of work and study such as business modelling and systems engineering. This approach is related to both the business process reengineering (BPR) and business process management (BPM) methods that began to attract attention towards the end of the twentieth century and into the twenty-first.

In this context, a business process can be regarded as "a set of partially ordered steps intended to reach a goal" (Feiler & Humphrey, 1993; Ertugrul, A. M., & Demirors, O., 2015). Other researchers provide more detailed definitions; Al-Humaidan, F (2006) cited both Davenport (1993) who describes the structure of a business process as "a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs: a structure for actions", and also cited Platt, D. G., & Blockley, D. I. (1994) who defined the business process as "the transformation of something from one state to another state through partially coordinated agents, with the purpose of achieving certain goals that are derived from the responsibility of the process owner". The business process defined in this way must be supported by rich business process modelling and implementation techniques that can support the achievement of organizational goals (Warboys, Kawalek, Robertson, Greenwood 1999).

Business processes may be classified into three categories: material processes, information processes and business processes(Medina-Mora, Winograd et al., 1992) . Material processes indicate human activities that are performed in the real physical world, while information processes are activities that deal with information flow and business processes deal with processing information. The business process related to the business domain must be formalized into proper framework to be investigated and modelled. Choosing the proper modelling tools and methods depend on the framework, and because of this the following section will formulate the business process framework.

2.3.1 Business Process Framework

A business domain consists of several business processes. Exploring the components and nature of business processes is an important issue in determining the methods required to model and implement them. This section will explore the business process framework, including a consideration of all characteristics of any method or approach required to model and implement a business process. Information system deals with different activities, where some can be computerised (implanted into software system) and some not. The work here wants to consider the most proper definition and framework to handle the business processes of any business domain in order to produce the software system, which leads that the above definitions cannot be considered for this research work.

Ould (1995) identifies three different types of business process: core, support and management processes. He also identifies the characteristics of the business process as

consisting of activities that are performed collaboratively, and as a cross-functional process which starts with an agent or customer. Similarly, Loucopoulos (2003) identifies the characteristics of a business process as consisting of activities, having products and customers, aiming to achieve a goal and having a horizontal form which crosses the boundaries of the organization.

Curtis, Kellner & Over (1992), who have dealt with business process modelling, have determined a conceptual framework for modelling the software engineering process and business process. They present the business process in terms of four views:

- a functional view, which represents the activities of the process;
- a behavioural view, which represents the ordering of activities;
- an organizational view, which represents the organization's structure and actors;
- an informational view, which represents the entities within the structure and the relationships between them.

Warboys, Kawalek, Robertson, and Greenwood (1999) which cited by Al-Humaidan, F (2006) stated that the business process can be defined from different viewpoints, which are the functional view, organizational view, behavioural view and informational view. The functional view deals with business process activities and information flow; the behavioural view deals with the timing of the execution of business process activities, and how they can be executed; the informational view deals with the informational entities required; and the organizational view focuses on who will perform the business process activities and where.

The framework by Curtis, Kellner and Over (1992) has covered certain issues of the business process, but the soft issues and the implementation have not been well-addressed, which are highly important if we need to produce a workable information system. This framework if adapted for modelling the business domain, must be modified to handle the soft perspectives of the business domain and the business processes included in it. The definition by Warboys (1999) and Curtis at el. (1992) identified the same views of business process into a framework, and as it argued above it need more tailoring to handle the soft issues of any business processes of any business domain. AL Humaidan (2006) consider these frameworks for business process modelling and this work also consider the both frameworks but after modifying them to be more holistic in order to handle the soft and implementation views.

Furthermore, Lochamp (1993) defines a business process as "a set of partially ordered process steps, with sets of related artefacts, human and computerized resources, organizational structures and constraints, intended to produce and maintain the requested software deliverables", while, (Johansson, McHugh et al. 1993) define it as "a set of linked activities that take an input and transform it to create an output", adding that "Ideally, the transformation that occurs in the process should add value to the input and create an output that is more useful and effective to the recipient either upstream or downstream".

This definition focuses more on the implementation while the other issues are related to modelling hard and soft issues in the development of the information systems. Al Humaidan F. (2006) defines a business process as something which "consists of related elements: ordered activities, constraints and business rules, human and computerized resources, a set of related artefacts, and organizational structure. These elements interact to achieve the organization aims and objectives". This definition is more related to the business process modelling and need to be more focused in order to deal with the development until a workable software system will be produced. Therefore, there is a need to acquire a holistic approach to handle all business processes within the business domain, which is one of the major issues considered by the present research.

To conclude this section, it is clear that for developing a useful information system, it is important to understand what they are for. An organization's business processes must be well defined and their implementation modelled. It is, however, difficult to be completely clear about what a business process is. The definition of a business process must deal with all soft and hard aspects of an organizational business process (Steve W. & Judith Hopkins, 2002; Al Humaidan F. 2006; Sewchurran & Petkov, 2007). This indicates that the definition offered by Warboys et al. (1999) and Curtis et al. (1992) are the most appropriate for the purpose of this thesis, and another views will be added to this definition to handle the soft and implementation perspectives of the business domain. These perspectives must be used during business domain modelling in order to reflect all business processes and other related artefacts. This definition, with its various perspectives, will be discussed and explored further in the 'Business Domain Modelling' section below. The modified framework is considered as a comparative framework to compare between DDD and SDDD in chapter 6.

2.4 Business Domain

2.4.1 Business Domain Modelling

The business domain model comprises of structural and behavioural parts (Bennett, 2007). The structural parts deal with the meanings of business artefacts and the business relationships between them, and the behavioural parts consider the business processes of the organization. The model includes all concepts to be used in modelling the business into a conceptual diagram, such as a class diagram. In agile software development, a domain model represents the application domain that facilitates communications between business experts and IT experts. Eric Evan's 'Domain-Driven Design' (2004) introduced 'ubiquitous language' as a communication tool between business experts and IT professionals. This is considered the backbone of the domain model. All concepts related to the design model are included in this language. During the creation of the business domain model, the rules of the business processes must be included and reflected in the model which will be used to develop the software system, and all views of the business process must also be depicted in the model. Further details about the domain model are explored in the 'DDD' section 2.4, part 2.

The business environment is not stable and this can affect the organizational business processes (business domain). It often forces business owners to set standards and methods to face different challenges in the market and to manage the business process life cycle. This must be supported by proper tools such as an information system to help in achieving their goals. It is reported in the literature that many information systems have failed because of several reasons, where poor business domain modelling is one of the most critical failure factor (Barjis, J., 2008). Factors include a tendency to concentrate on the IT technology, rather than on understanding the business processes involved and modelling them to create a rich business domain model. An exploration of the business process literature reveals that there is no existing methodology that can deal with an organizational business process in a way which can facilitate and manage the development of its lifecycle (Al Humaidan, 2006). There is a need for a methodology that can handle all soft and hard aspects of the business process (Steve W. & Judith Hopkins, 2002; Al Humaidan F. 2006; Sewchurran & Petkov, 2007). Al Humaidan's (2006) work considers this issue and proposes a framework for business process modelling as a workflow system. However, considering workflow alone will not deal with all issues related to the business domain processes, since this approach concentrates on business processes alone, rather than the whole business

domain. The business domain is more comprehensive and includes all processes with related services and other artefacts required to implement the software system. In addition, Al Humaidan's (2006) work ends with a model of a workflow system and does not progress to implementation. This raises the issue that a comprehensive framework is required to facilitate the process of investigation, modelling and also implementing the business process to achieve organizational goals. The existing methods and methodologies of business process modelling deal with specific aspects only. Aguilar-Saven, R. S. (2004) investigates some of the methodologies of business process modelling, while (Kettinger, Teng et al. 1997) investigates different methodologies of business process reengineering. These methodologies concentrate on the modelling of business processes, as discussed before, but not on the modelling of a business domain.

The following sections will review information systems development methodologies and multimethodology frameworks required for business domain modelling and implementation, and in part2, section 2.4 will focus on DDD as a dominant approach among these frameworks.

2.5 Information Systems Development Methodologies and Tools

2.5.1 Definition of method and methodologies

The literature has presented several definitions of methods and methodologies, with no clear distinction between the two. Either one of these terms have been used in most of the existing studies. According to Avison & Fitzgerald (1988) a method or methodology is a "recommended collection of philosophies, phases, procedures, rules, techniques, tools, documentation, management and training for developers of Information Systems". However, few researchers have provided a difference between the two, where methodology is considered as a more comprehensive concept than a method that is utilized logically for evaluating the adequacy and reliability of a method. A method, on the other hand, is a manner with which a task is completed (Checkland, 1981; Vonk, 1990).

A methodology comprises of three essential components, which are, a work breakdown structure for providing a systematic procedure of executing processes along; techniques to implement those processes; an advising on handling the quality of the results. A methodology is a process that depends on various elements such as the human resources (technical staff, management team) and material resources (software and hardware tools). The aim of a methodology is to incorporate changes efficiently in the systems via controlling

all the operations. For this purpose, the methodologies must be prepared in an understandable manner, which can be transferred and learned over different development scenarios (Checkland, 1981).

2.5.2 Definition of information system development methodology

To develop an information system, information systems development methodology (ISDM) must be used to ensure that all the system's perspectives are achieved. ISDM is a notional theory of practice for the information systems development process, and is used by information systems developers as a guide to the process of intervention into the information systems environment. Information systems development methodology can be defined in different ways. Fitzgerald (2003) defines ISD methodology as "a systematic approach to conducting at least one complete phase of information systems development, consisting of recommended collection of phases, procedures, techniques, tools, and documentation aids". An information systems development methodology incorporates a world-view, models, methods, techniques, management and training into a coherent theory to guide the practice of information systems development (Michaailescu, D., & Mihailescu, M. (2010). The world-view associated with the methodology is due to the influence of the methodology's author. However, if the developer and the author are not the same person, the world-view of the developer will also influence the methodology and its use. The methodology may or may not be made more efficient with the aid of technology. The views of business experts are not the same as those of technical people, and this can lead to communication difficulties between the team members responsible while the information system is being developed. Therefore, it is important to follow a methodology that can facilitate the process of information system development.

There are different approaches to classify software development methodologies. Information system development methodologies can be grouped into soft and hard methodologies. Hard approaches are originally developed from systems engineering, where additional activities emerged from industry. Soft approaches came from outside industry by Peter Checkland, 1981 who developed SSM in Lanchester University, UK. SSM succeeded as a general purpose-problem solving methodology to handle the messy or unstructured problems. One classification approach has classified hard methodologies into traditional approaches (heavyweight) and Agile approaches (lightweight) (Boehm & turner, 2003; Charvat, 2003; Highsmith, 2001; Wysocki, 2009). Heavy weight like Waterfall (Benington, Herbert D., 1956, 1983), Iterative Waterfall (Winston Royce, 1970), Waterfall (Bell, Thomas

E., and T. A. Thayer. , 1976), B-Model (Birell and Ould, 1985,1988), Information engineering (Martin & Finkelstein, 1981), Spiral model (Barry Boem, 1988), SSADM (Eva, 1994), Unified Software Development Process (Jacoboson, Booch, & Rumbaugh, 1999), prototype model (Pressman, 1994), and Microsoft Solution Framework (MSF) model (Microsoft, 2004). Other classification approach classified hard approaches into models based on sequential approach like waterfall model, and models based on iterative approach like prototype model, spiral model, unified process model, Microsoft solution framework, and agile methods (Predrag Matkovic & Pere Tumbas, 2010).

Other approach has classified hard approaches into structured methodology and object oriented methodology. The object-oriented methodology deals with modelling the problems into abstraction in order to be implemented as a software systems. Bennett et al, (2002, p57) continued to assert that object orientation can model complex information systems through the conceptual diagrams, and it became a necessary approach to deal with the system complex requirements. Object oriented approach breakdown the complex system into small subsystems with less complexity and support the re-use of IS development models and program codes.

These issues encouraged the development of agile methodologies which are a combination of soft and hard approaches. Agile methodologies were reviewed in separate section of this chapter.

2.5.3 Hard Problems vs Soft Problems

While selecting a methodology to solve a problem, a distinction between hard and soft problems must be considered to guide the selection. Hard problems are considered well defined in answering the question 'how'. With hard problems, there is a solution by which the aims are achieved. Hard approaches to system development have been succeeded in developing information systems from the technical perspectives. Curtis, 1998 show that information system sometimes not accepted by user as a solution of spurious problem. This raise the issue that an alternative approach is required to capture the human elements (soft) of a system.

Avison & Taylor, 1997 have classified the information systems problem situations into five classes, depending on requirements structure, problem definition, and their level. They determined the following five classes of problems:

- Well-structured situation, well defined problem, and clear requirements.

- Well-structured situation, clear objectives, and uncertain user requirements.

- Unstructured situation and unclear objectives

- High user interaction with the system

- Complex problem, required contingency approach to ISD, and combining two or more classes 1-4.

The soft (unstructured) problem is concerned with 'what' and 'how' questions, and according to Checkand, 1999, 'hard' and 'soft' approaches are different in nature and the main difference between them is that the hard system thinking is suitable for well-defined technical problems while soft system thinking is more suitable for fuzzy(unstructured) situations which include cultural and human being issues. According to Avison & Taylor, 1997 hard approaches are applicable for classes 1&2 problems while soft approaches is applicable with classes 3,4, and 5. Each approach has its strengths and weaknesses. Harry, 1994 compared between hard and soft problems as follows:

Hard Problem	**Soft Problem**
Defined	Undefined
Clearly bounded	Fuzzy-edged
Separable problem	What is the problem?
Clear who ought to be involved	Not sure who ought to be involved
Information needs known	Unsure what information is needed
Know what the solution would look like	Not sure what the solution would look like

Table 2-1: Distinctions criteria between hard and soft problems

Based on the above distinctions between hard and soft system approaches, the following sections will review the related works to both approaches in order to show the applicability of each methodology belonging to one of the approaches for handling the soft and hard system issues.

2.5.4 Hard system development methodologies

Tradition methodologies or (heavy weight) and agile methodologies or (Light approaches) are classified under hard system development methodologies. Under each category, there are different types of methods or approaches. The following review of the related work will critically analyse both the ISDM categories (tradition methodologies & agile methodologies) to address the problem that one reason of information systems failures came from the ISDM because of the weaknesses of handling the system perspectives 'hard' and 'soft' and to show that one methodology cannot handle all the perspectives. Also this review will try to explore the multimethodology work in order to find their applicability to handle all perspectives, and then help to eliminate IS failures.

2.5.4.1 Tradition Methodologies (Heavy weight approaches)

Information systems development methods have been used for many years; indeed, since the 1970s. One of these hard methods is the 'Structured Systems Analysis and Design Method' (SSADM), which was developed by Ashworth and Goodland in 1990. However, this method came from civil and mechanical engineering and is not popular with programmers. The reason behind this is that the method places considerable emphasis on planning and a lot of time must be spent on it before anything is produced. This approach focuses on developing certain models to construct the information system. From a management perspective, this approach is good because it allows them to plan and predict the schedule and budget for the system development. However, it may be argued that because this approach requires the project manager to plan a lot of the work and activities involved in the system's development, this will take a lot of time and then there may be problems in making any changes to what has been planned. The following are some of the traditional methodologies utilized in the development of information systems:

1. Waterfall

Waterfall is first introduced by (Benington, Herbert D., 1956, 1987) and modified by (Bell, Thomas E., and T. A. Thayer. , 1976). Waterfall is a linear framework that comprises of

sequential steps for developing an information system (Figure 2-1) (Adenowo & Adenowo, 2013).

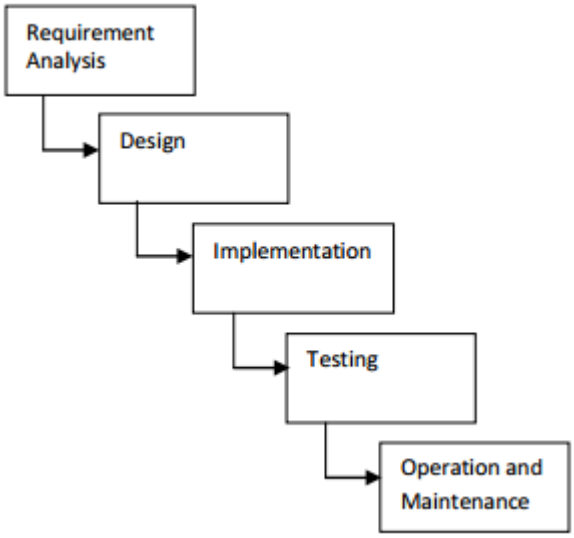


Figure 2-1: Waterfall methodology for ISD

The waterfall methodology is segregated into different phases, where each phase is executed in a sequential manner and cannot be re-visited again. Specifying the requirements of the business is the first phase. The requirements form the basis of the information system, after which analysis is conducted and system is designed. After the designing and development of the system, it is implemented via coding and then tested via unit and integrated testing to check the proper functionality of the system. The last phase addresses the operability of the system and inculcates further changes as and when required, thus maintaining the system. There is no overlapping between these phases (Adenowo & Adenowo, 2013).

2. Iterative Waterfall

Iterative Waterfall is introduced first by Winston Royce, 1970. This is a prototype framework that breaks the process of methodology in smaller sections for easy execution of development process (Figure 2-2) (Munassar & Govardhan, 2010).

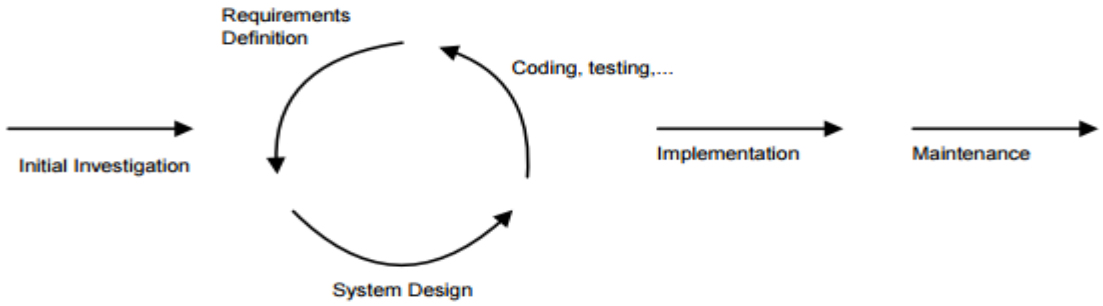


Figure 2-2: Iterative waterfall methodology for ISD

This model is similar to the waterfall model, however, unlike the traditional waterfall, here the phases can be re-visited during the development of the information system. After identifying and specifying the requirements and designing the system, the requirement phase can be re-visited if there are certain changes in the dynamic market that have further changed the requirements. As user-specific needs are never constant, this model is effective in addressing the changing requirements as and when possible. It is also useful in resolving ambiguous objectives through iteration and provides flexibility in the designs (Munassar & Govardhan, 2010).

3. Spiral Model

The spiral model is developed by (Barry Boehm, 1988). The spiral methodology is a combination of both the iterative and linear approaches that is efficient in developing an information system (Figure 2-3) (Munassar & Govardhan, 2010).

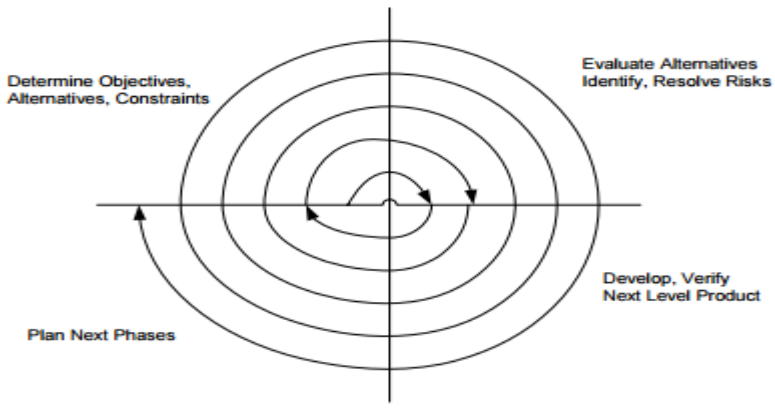


Figure 2-3: Spiral methodology for ISD

The methodology starts with the identification of the objectives and alternatives, evaluation of alternatives and risk management, development and verification of the system and planning the next iteration. Unlike the other models, the emphasis is laid upon the evaluation of risks and their effective mitigation. An information system development process repeatedly follow these spirals (iterations) until the required system is developed. High amount of risk analysis is performed in this model, which may also lead to higher cost (Munassar & Govardhan, 2010).

4. B-model

The B-model is developed by (Birell and Ould, 1988). The B-model is an extension of the waterfall model that ensures the constant improvement of an information system (Figure 2-4) (Ruparelia, 2010).

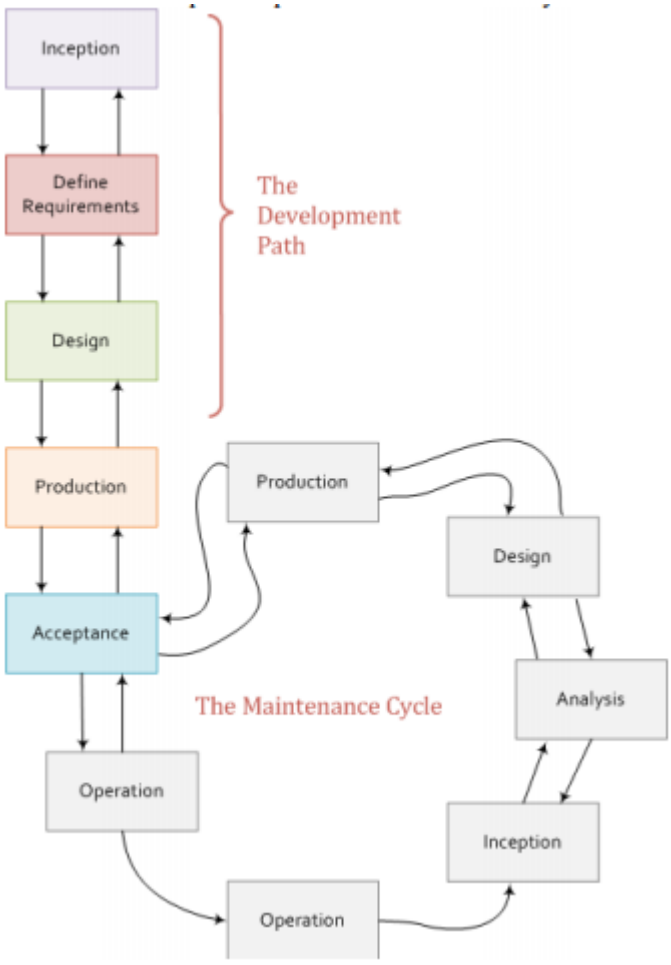


Figure 2-4: B-model for ISD

This model was developed to ensure that new systems could be effectively inculcated in the existing information systems. The model separates the development cycle of the ISD process, wherein the system is developed by following the similar steps of requirement specification, designing and implementation and testing from the maintenance cycle, wherein the information system is maintained after its development by following the similar cycle (Ruparelia, 2010).

2.5.4.2 Agile Methodologies (Light Approaches)

Systems specialists and developers therefore started looking for simpler approaches and methodologies as alternatives to SSADM and other hard approaches. Agile methods were introduced, which were declared to be a solution for this situation. These aimed to provide sufficient processes for any given project but tried to avoid detailed descriptions of processes. Agile methods or 'light approaches' received more interest during recent years as a compromise solution between heavy weight methods and no development process, providing enough process for any given project (Ambler, 2002). These methods influenced by object-oriented programming languages and object-oriented and relational databases. These methods support the programmers to develop fast solutions and to avoid them going through detailed design and development steps.

Different researchers define the term 'agile' in different ways. Alistair Cockburn, the first one introduced agile method and defined it as:

> "agile implies being effective and manoeuvrable. An agile process is both light and sufficient. The lightness is a means of staying manoeuvrable. The sufficiency is a matter of staying in the game" (Cockburn, 2002).

There are different agile methods s, which use UML with varying degrees of agility, such as 'Unified Software Development Process' (USDP) (Jacobson et al., 1999) and the 'Rational Unified Process' (RUP) (Kruchten, 2004; Manalil, J., 2011)), both of which have attracted a the attention of developers. Also there are other important agile method like Alistair Cockburn's agile method called 'Crystal' family of methods (Cockburn, 2001), Jim Highsmith's 'Adaptive Software Development' methods (Highsmith, 2001) and 'Feature Driven Development' (Jeff De Luca,1997;1999).

Agile methods can be modified and changed from project to project. So they classified as a project base methods. The modification can be done by comparing conceptual model of the system by its requirements by considering all the related issues such as the cultural requirements. With heavy weight methodologies, a lot of time spent on requirement configurations and to get the customer to 'sign off' before moving to the design and implementation. This approach is not working well since the business requirements can be changed and not stable. This problem caused clients to go for agile methods since requirements will be depend on the model base and it can be modified as a learning take place through the project phases. Without stable requirements, a predictable plan cannot be achieved. This raises the question of how some degree of control may be exerted over such unpredictability. The new development methods focused more on 'use cases' and 'iterative' development techniques. Use cases were discussed in the UML section is a piece of functionality that can support user understanding and provide them with meaningful value.

In an iterative approach, developing a system consists of short projects called iterations. The output of iteration will be tested and then all iterations will be integrated into the whole system. However, it may be argued that there are different types of project where requirements are so unclear (complicated business processes). For such projects, the use case approach is not suitable for identifying the right iterations. For this reason, this thesis believes that techniques from soft systems methodology (SSM) should be added in order to explore the business domain clearly and provide structure to the situation. This explores the idea that dealing with business processes from the business domain perspective will contribute to developing an understanding of the system requirements which can directly reflect the business domain. Such an approach will allow different stakeholders to converse in a similar language, thus improving their understanding of the requirements involved in building the business domain model and implementing it as a software support system. This view is presented in the domain-driven design framework (Evan, 2004). An agile software development methodology fits well here because they focus on the business values while DDD concentrate on the business model to be aligned with the software system. DDD followed an iterative approach while agile methodologies such as SCRUM or DSDM offer a project management framework. To manage a DDD implementation project, a combination of XP (develop a software system), and SCRUM (project management framework) is advised to be used. This research aims to make further improvements to these methodologies by developing a new approach which combines SSM with DDD.

One of agile methodologies is 'Extreme Programming-XP' which emphasises on iterative and incremental development methods and provides explicit and hands-on methods for developers. Therefore, extreme programming and domain-driven design are a perfect fit for each other. There are no conflicts between the values of these two development models, and while XP is more practical, DDD is more philosophical (Oqvist, 2011). This argument has encouraged the adoption of DDD as a base approach for the proposed SSDDD, since DDD is close to agile methodologies.

Another agile methodology is 'Feature-Driven Development-FDD' which is developed by Jeff De Luca (1997). FDD is a management-supporting tool that suggests a specific framing of the process as well as iterative development, but does not provide guidance in respect to specific development methods. Other agile methodologies were discussed in the literature, but it is clear from those reviewed in this section that these methodologies focus on making the development process shorter than traditional hard approaches. However, none of these, nor any of the others, have tried to solve the problem of soft system aspects. This supports the goal of this research, which is to combine methods and techniques from different approaches. Many of these methodologies, such as RUP(Rational Unified Process) by (Kruchten, P., 2004) and (Manalil, J., 2011) and USDP (Unified Software Development) by (Jacksbon,1999), adopt UML as a modelling approach, which has encouraged the proposed combination of UML with SSM, since it can be utilized to handle the soft aspects of the system being developed.

2.5.4.3 Related Frameworks

1- Multiview Framework

Avison and Wood-Harper (1990), developed a multiview methodology for ISD, wherein the development process comprised of multiple players. The basic concept of this approach is that information system development is an integrated process where the developers design and implement the system for the end users by deploying a particular methodology. Both the soft and hard aspects of building the system are incorporated by working in alignment with the soft system methodology and Yourdon Systems Modeling (1989). The authors have developed and modified the framework by using action research method in an academic setting and comprehends different perspectives and views. However, the framework is not applicable in all the situations. The multiview methodology is segmented into several phases, as mentioned below:

- Assessing human activities
- Evaluating the information
- Analysis and design of socio-technical aspects
- Design of the human-computer interface
- Design of technical aspects

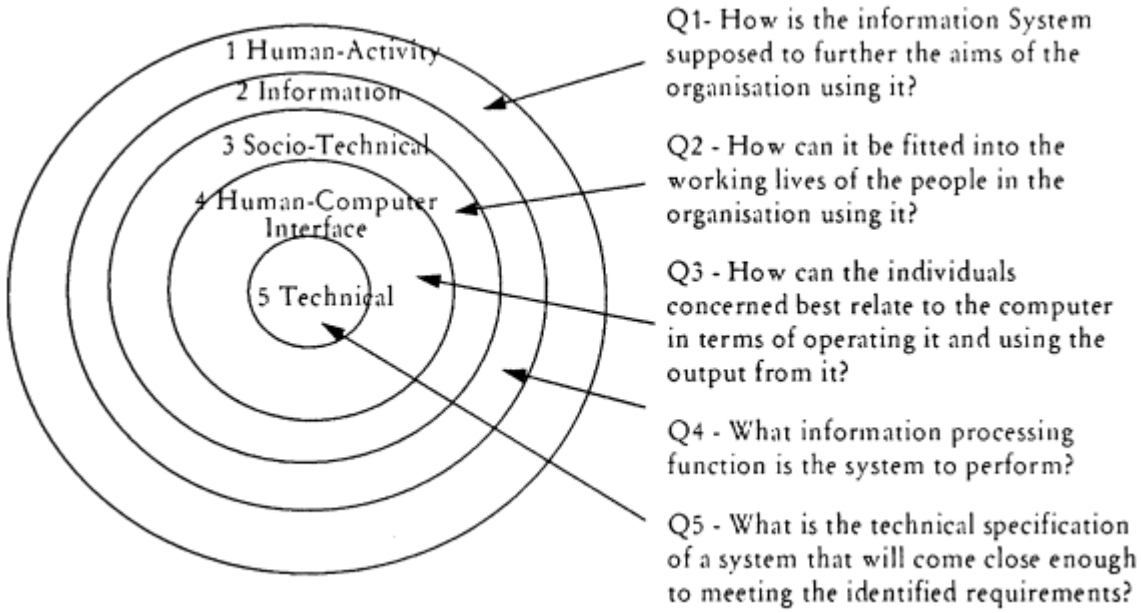


Figure 2-5: Multiview Framework

The above mentioned phases address to five different perspectives, which is why the framework is referred as multiview. The model offers a progressive development of the information system to satisfy the user requirements, by addressing both the technical and human terms. The outputs of each phase can either be fed to the next phase or work as a separate output. As per the organizational needs, any order of executing the framework phases can be followed, while also removing a phase altogether. However, the process of how to jump from one phase to another, in case where one or two phases are to be omitted, is not determined by the framework. Also, the framework is unable to provide the tools and techniques that can be used to develop the information system, and it is not easy for the whole stakeholders to deal with this methodology specially the business experts because of the difficulty to understand the technical parts.

2- Soft Workflow Modelling (SWM)

Al-Humaidan (2006) developed a framework that aimed at comprehending different perspectives of hard and soft requirements. The model, soft workflow modelling, was developed for the workflow of organizational business processes. The researcher has incorporated both SSM and UML for this purpose, where the focus is laid on SSM, with UML covering the aspects that SSM cannot.

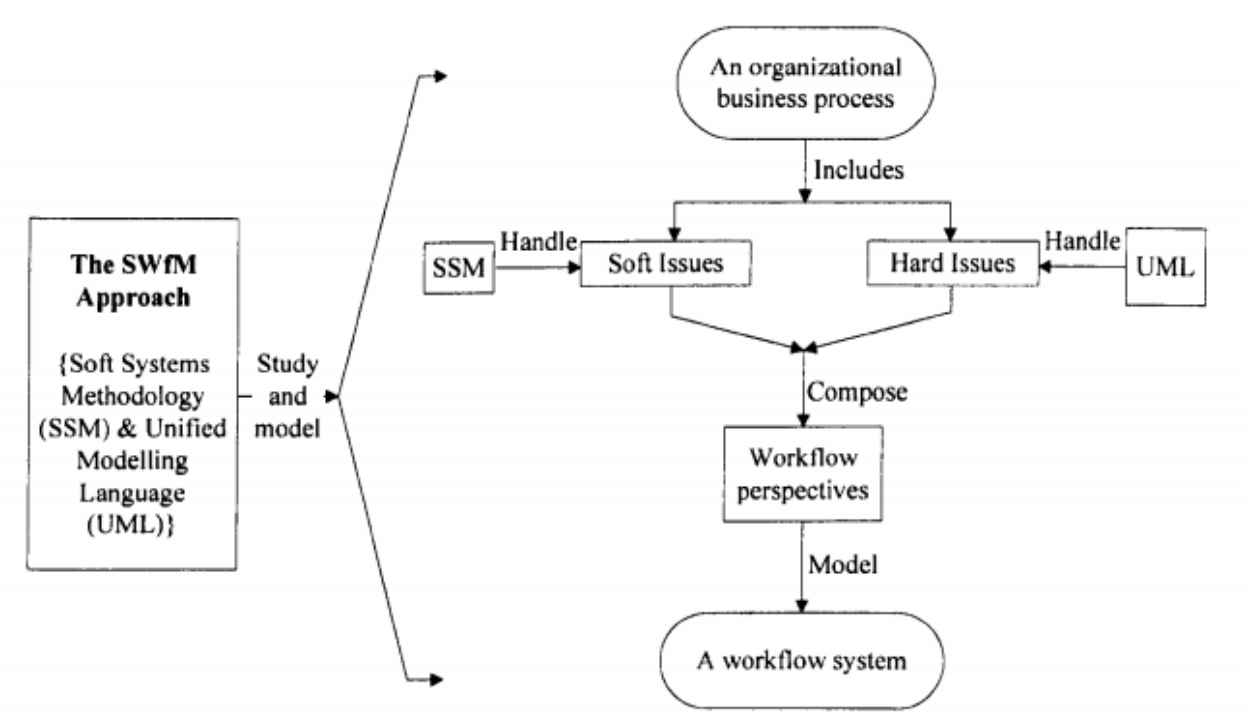


Figure 2-6: SWM Framework

The soft system methodology, in this framework, evaluates the organizational business process and investigates whether or not it can be modelled as a workflow system. The issues that are not handled by SSM, are addressed by UML. The UML looks into the tangible and technical elements, while the SSM addresses the human aspects. However, the framework addresses only two major concepts, which are organizational business processes and workflow system modelling, the rest of the phases have to be managed efficiently to

gain maximum benefits. Also, the approach is not evaluated or verified using real scenario case study, thus imposing a limitation of its actual implementation.

2.5.5 Integrating SSM and UML

Chechland, 1981, 1999 mentioned that SSM is an approach to business process modelling that can be used for both general problem solving and management of change, and it has been most successful in the analysis of complex situations where there are different views about the problem definition (i.e. 'soft problems'). SSM supports the business improvement by developing systems models and the activities that must be performed by an organization to achieve their goals, while UML modelling (use-case) is a requirements engineering technique to identify the system activities, but these activities are driven from the systems users rather than the system itself.

There are different previous efforts to integrate SSM with other hard approaches like structured development methods done by (Keys, P. and Roberts, M., 1991; Lewis, P., 1995; Miles, R., 1992; Mingers, J., 1988; Prior, R., 1990. Later on, some efforts to integrate SSM with object oriented were made, (Bustrad, et al, 2000; Steve W. & Judith Hopkins, 2002; Al Humaidan, 2006; Sewchurran & Petkov, 2007) which executed the integration of SSM with UML use cases. Integrating SSM with UML is an increasing approach and the work in this area is essential to determine the requirements specification and the identification of business processes. The work in this area demonstrates the importance of such integration for investigating a complex and messy problem situation. Using techniques from hard approaches alone (e.g UML) is not applicable when the requirements unclear and the combination between SSM and UML is required to evaluate the requirements from the perspectives of different stakeholders. Therefore, the business processes will be constructed in the minds of stakeholders. These researchers illustrated that the combination of SSM and UML encouraged the SSM exploration of system activities from the system itself and their conversion into use cases (representing the system activities) from the users' perspectives (Bustrad, et al, 2000). Combining UML with SSM might help in modelling both 'hard' and 'soft' system aspects of the business domain to develop IS, which are expected to reflect business needs (Al Humaidan, 2006; Sewchurran & Petkov, 2007; Bustrad, et al, 2000; Steve W. & Judith Hopkins, 2002). This combination is achieved using use case diagrams that will accommodate all the knowledge generated by SSM conceptual models during the business domain investigation phase. Then, the transition from the business domain model (SSM conceptual diagram) into UML use cases will start. After that, UML

diagrams will be modelled using use case diagrams such as the class diagram, which will represent the main diagram of the business domain model. Tools from the object-oriented domain (UML), such as class diagrams, activity diagrams, sequence diagrams and interaction diagrams, have proved to be accepted as modelling tools for modelling business processes (Fowler & Scott, 2000).

Recent research work shows that the combination between SSM and UML is used to contextualise the problem space using SSM and developing UML models to solve the complex problems (Ross Fenning et al, 2014) to design a complicated search engine for BBC (British Broadcasting Corporation). This recent research effort is fit with what proposed and done in this thesis work and encouraged the continuation of this research direction.

Other recent works have presented systems thinking-based approach for finding the requirement in complex situations, by exploring and identifying the challenges of complex situation requirements gathering to be the requirements nature, the observer role, and the system environment (Polinpapilinho F. Katina, Charles B. Keating, Ra'ed M. Jaradat, 2014). These researchers focused on systems thinking approach as a holistic approach for systems requirements gathering and to consider the system soft perspectives since the system in a complex environment situation. This work is fit with what the thesis proposed of mixing different techniques to handle the complex system situation.

Minger (2001) added that gathering understandable, consistent, modifiable, and verifiable requirements is difficult with the complex situation. Further, to achieve such requirements, a change in paradigm is required such as an integrated multiple infrastructure through holistic thinking, as done in this thesis to mix different methods from different paradigms to deal with complex situation (Minger, 2001). This thesis work adapted Mingers work and considered mixing of different systems development techniques from different paradigms.

Galvin and Lane (1999) have mentioned that transiting from SSM to UML use cases imposes a problem as these methodologies are based on different paradigms ('soft' and 'hard'), and will be difficult for mapping the information gathered by the first methodology to the other one. SSM is interpretivist while UML is a subjectivist approach (object-oriented (OO)). Using the objectivist approach through OO modelling methods, the existing problem hard issues will be handled technically using different UML diagrams, while other soft issues relating to the organization culture and politics will be missed and this will lead to non-complete information system. The solution suggested here is informed by Mingers' (1997; 2001) work

on multimethodology or plurality, which is used to show the crossing of positivist and phenomenological paradigms (SSM versus hard object design methods) to solve the problematic situation by considering the right actions to do that.

This research considers the practice of combining SSM and UML methodologies, a practice which may also be referred to as methodological pluralism or multi-paradigm intervention and research. Sewchurran and Petkov (2007) argue that their work on mixing SSM and UML is different from past attempts at combining methods, since it is better justified methodologically as multi-method research in systems thinking and operations research Mingers (2001), and also because it is formulated as an action research approach. Sewchurran and Petkov (2007) state that SSM plays an organizing role in their proposed framework, so such a combination of methods may be considered an enhancement of the multimethodological possibilities discussed and justified by Mingers (2001). This thesis argues that the difficulties highlighted by Mingers (2001) about mixing methods from different paradigms can be avoided through the separation of activities within the SSM and UML parts of the proposed framework (Salahat et al., 2009). This research therefore considers the transition from the CPTM of SSM to UML use cases as it's considered by other researchers before and a new elaboration technique is developed to do this.. The results from one stage can be continuously used as an input to the next stage in the action research project, and this involves a number of stakeholders. It is argued that, through this adoption of an action research approach, the difficulties expressed by Mingers (2001) can be avoided.

SSM conceptual model was used to model the activities of the business domain that affect the business as a whole, while use cases are concerned only with activities that can be directly supported by a software system. After presenting and reviewing different transition methods, an appropriate method is recommended for use in this research. The following section explains how the transition point may be identified, followed by a review of the transition methods discussed in the literature.

2.5.5.1 Identifying a transition point

There are different techniques and tools utilised by SSM. For the transition purpose from SSM to UML, it's important to identify which technique can support this. For this work use case was considered the most suitable tool to be used for this transition which will support the development of the software support system for the investigated domain later on.

Galvin and Lane (1999) described the process of moving from SSM to OOA to be a top down to explore the business domain. They were considered the use case description and diagram is the more appropriate to handle this process. Figure 2-7 represent the transition process (Galvin & Lane, 1999).

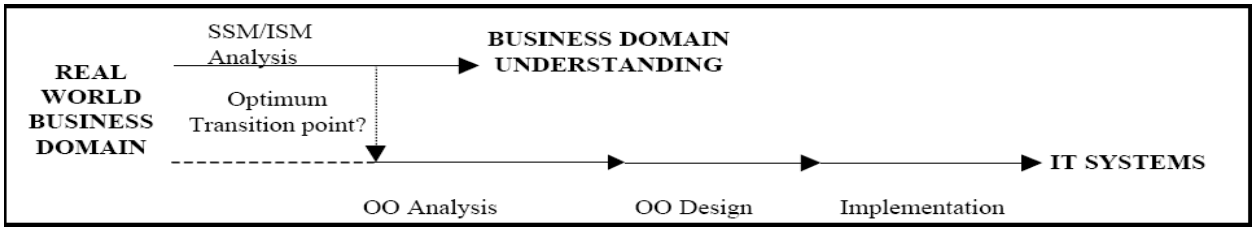


Figure2- 7: SSM to OOA Path

This thesis has reviewed the linking of SSM and UML and the transition methods identified in the literature. One of these methods has been selected and elaborated for use in this research, and the revision of this transition method is presented in the following section.

2.5.5.2 Review of transition methods from SSM to UML use cases

Different efforts have been made to link SSM and OOA. For the context and focus of this research, the linkage of SSM models and UML use cases will be considered. Galvin and Lane (1999), in their work for the UK Ministry of Defence, identified four transition methods, two of which considered the transition to UML use cases. The first method is to derive use cases from the root definition, while the second is to derive them from Brian Wilson's (1990) conceptual primary task model (CPTM). These two transition methods are presented below.

1- Extracting use cases from root definition

This method consider the extractions of objects from the root definition (RD) which yields a few objects to build the object model. This method consider the root definition as a point to start the business domain investigation using OO approach. This method of transition consist of the following stages:

1- Start with root definition to identify the purpose and main usage of the system then to identify a high level use cases set.

2- This required the Business Domain experts to be involved while developing the use cases set.

3- After this, the process will continue using OOA tools and techniques.

According to Galvin and Lane (1999), the advantages of the method are that the OO analyst not necessary to be professional in SSM; since he will depend on the root definitions as one major output of the SSM; and the utilization of use cases give a chance to describe the business domain in details to be used for developing OO models. At the same time, they highlight the disadvantages of the method to be an indirect transition from SSM which is required detailed analysis of use cases, may be some of use cases not recognised from the root definition, and since the extraction of CMs from RD is go as an iterative process which cannot depend upon to extract use cases. Galvin and Lane (1999) state that the advantages of this transition method are not enough to ignore the disadvantages, and therefore this method is not suitable to be considered as a transition approach from SSM to UML.

This thesis agrees with this assessment, since the root definition is still being used to construct the CMs as an iterative process, and so it cannot be depended upon for the extraction of use cases.

2- Deriving use cases from the CPTM

Galvin and Lane (1999) identify eight phases/stages to make this transition. These stages were presented in table2-2 (Galvin and Lane (1999).

According to Galvin and Lane's (1999) assessment, this method is better because it is a natural transition and no paradigm shift in the modelling approach. In addition, they highlight that the transition depend on the rich knowledge gained from SSM which represented by different conceptual models including all information required to perform different activities. So, the conceptual models represent a standard framework of the business domain to develop the use cases then continue until system implementation. At the same time, they express that this is an indirect transition and they were considered this as a disadvantage of this transition method because it requires detailed use case analysis to understand the real business domain problematic situation.

This research agrees with this assessment, since the eight steps to be followed may be considered a lot of work to be done, particularly in terms of use case analysis. However, an elaborating technique has been proposed (Salahat et al., 2009) to enhance this point and to make it easier for developers of IT systems. This technique is explained in Chapter Four and will be covered through the illustrative 'Peer-Tutoring' case study in the following section.

a. Scope and prioritise the activities for the project to be investigated for possible IT support and use case modelling. b. Identify the OOA tools to be used for this conversion, and here it's use case method. Then determine those activities need to be supported by IT (to be computerized), those will not need IT support(not to be computerized), and those required to be decomposed into smaller activities. c. Actors of the smaller activities need to be identified, and CPTM can be used in this stage to assist the identification of actors. It is not necessary for actor to a person, but it can be anything to perform the activity. d. High level use cases to be developed in this phase. The smaller activities will be the names of high level use cases which are used in the transition from SSM to UML use cases. Objects belong to each use case will be named by underlined nouns. e. Multi-level use cases to be developed in this stage. The high level use cases will be decomposed into low level use cases. f. Top level objects need to be identified, in this stage, from use cases through extracting the nouns and show the relationships between objects. g. Top level services required will be mapped onto objects. Objects will be mapped to business processes and activities involved in the business object model. This will help to identify any missing objects. h. Continue analysis employing conventional OOA/D techniques (e.g. OMT with UML notation).

Table2- 2: Stages of transmitting from CPTM to use cases.

Therefore, the CPTM transition method, combined with the elaboration technique mentioned above, was selected as the best method to use since it accommodate all the relevant stakeholders' viewpoints. This method not only for converting CPTM to use cases, but it can be used to convert individual conceptual model to use cases also.

Different IS publications presents many efforts of different researchers whom tried to combine SSM with other methods to help the developer to determine an improved requirements for information systems development (see Mingers, 1995; Bustard, Dobbin & Carey, 1996; Wade , 2004; Al-Humaidan & Rossiter, 2004; Stowell, 1995; Wilson, 2001, and others). They discuss the benefits and concerns about how techniques from two

philosophical backgrounds may be linked without negating the advantages of each individual technique (Mingers, 1992). Mingers (1995) agreed that SSM and ISD he mentions that this will not be a serious problem since there must be a transition approach which lead to a concreteness and result in action being taken.

Bustard, He, and Wilke's (2000) work presents an effort to link SSM with use case analysis. However, their work does not distinguish between architectural modelling, analysis models and design models. In addition, they do not express the difference in ontological assumptions between SSM and use case analysis.

Similar to the approach presented in this research, Al-Humaidan and Rossiter (2004) propose the use of the conceptual primary task model (CPTM), and the direct mapping of each activity from the CPTM to a use case, as proposed by Galvin and Lane (1999). However, the research reported in this paper assumes that a use case is a specific use of a system that is part of a business process. A CPTM is more likely to map to a business process rather than to a specific use of the system. Al-Humaidan and Rossier (2004) refer to UML modelling taking place within SSM, but there are no further details provided about how this idea is implemented or formalized. This research work has considered this point and also adopts the view that SSM is the guiding methodology and all UML modelling techniques are embedded within it (Salahat et al., 2009). In addition, an implementation pattern is embedded within SSM to implement the modelled system, which other, previous efforts at combining approaches have not done. The conversion method adopted depends on moving from CPTM to use cases through an elaboration technique presented in Chapter 4.

2.5.5.3 Peer-tutoring illustrative case study

Through this thesis research work, the transition method from CPTM to UML use cases is considered to be the most suitable transition approach, and this is applied through the elaboration technique presented here and in Chapter 4. The proposed SSDDD framework is explained through a peer-tutoring case study which is used to illustrate the conversion method from CPTM to use cases as reviewed in this chapter. Joseph Ucizi Mtenje (2010), a postgraduate student in the Department of Informatics at the University of Huddersfield, selected the peer-tutoring system as a project to be developed using the SSDDD framework. Through his work, he evaluated the transition method from CPTM to use cases

using Galvin and Lane's (1999) approach with the elaboration technique proposed by this research (Figure 2-8). This previous research work (Salahat et al., 2009) defined use cases as abstractions of business activities which can be used to model the business domain model using UML diagrams through the philosophy of DDD which emphasised on the idea of 'Knowledge Crunching' during the different phases of transition. By combining different developed SSM conceptual models, anew diagram called the consisious primary task model (CPTM) will be generated and used to map human activities to UML use case diagrams using the new elaboration technique proposed by this work (Salahat et al., 2009). The following figure (2-8) presents this technique:

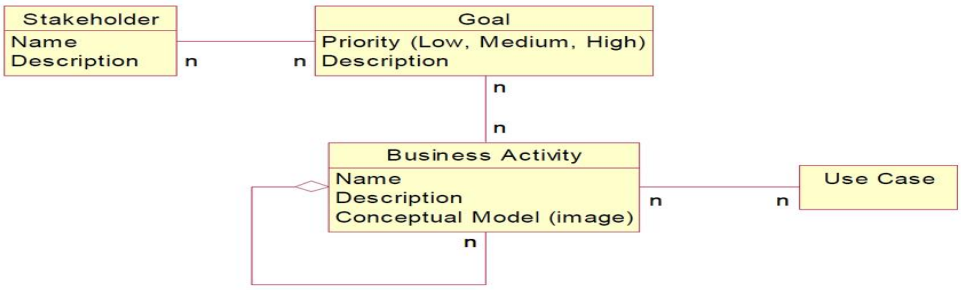


Figure 2-8: Elaboration Technique of Transition from Conceptual Model to Use Cases

Using Galvin and Lane's (1999) approach and the elaboration technique presented in Figure 2-8, the transition from the CPTM of the peer-tutoring system to use cases is presented by the supervised postgraduate student Joseph Ucizi Mtenje (2010) as part of his final project which lead to the application and evaluation of the proposed framework SSDDD as a software development approach . The complete application and evaluation were presented in chapter 5. This is included here to demonstrate how the selected transition method can work with the proposed elaboration technique. Joseph Ucizi Mtenje (2010) mentions that the this transition method from SSM CPTM diagram to UML use case diagrams is preferable to other methods because it covers all stakeholders' viewpoints, and therefore deals with all the requirements presented by stakeholders through the root definition phase of the SSM application process. The phases described below are those discussed by Galvin and Lane (1999) regarding the process of conversion, combined with the elaboration technique which focuses on starting with the stakeholders. Joseph Ucizi Mtenje (2010) applied this transition approach and identified the following phases:

Phase 1: Peer-Tutoring System activities scoping and prioritising. The activities of the conceptual models representing PTS were selected, prioritised, and presented in table 2-2.

1- Add tutor	7- Select room	13- Apply policies and procedures
2- Add tutee	8- Schedule sessions	14- Measure success of system
3- Add room	9- Mark attendance register	15- Increase programming skills
4- Add module	10- Test the understanding of tutees	16- Redeem rewards
5- Select tutor	11- Allocate rewards for tutors	17- Reduce workload
6- Select tutee	12- Reward tutors	

Table2- 3: The prioritised activities of PTS

Phase 2: The scope of UML to be identified. Low level activities will be decomposed or combined and then use cases will be extracted from them. This will be done for those computerised activities only, while other non-computerised activities will not be converted into use cases. Table 2-4 represent those activities involved in the transition process.

1- Add tutor	5- Select tutor	9- Schedule sessions
2- Add tutee	6- Select tutee	10- Mark attendance
3- Add room	7- Select room	11- Allocate and reward tutors
4- Add module	8- Select module	12- Update diary

Table2- 4: PTS activities involved in transition

Phase3: Identify actors to perform the activities identified. The following actors (table 2-5) were identified at the stakeholders' definition stage:

1- Tutors	2- Tutees	3 -Management	4-Lecturers

Table2- 5: Actors of PTS

Figure 2-9 shows the actors linked to their respective activities (Joseph Ucizi Mtenje (2010)).

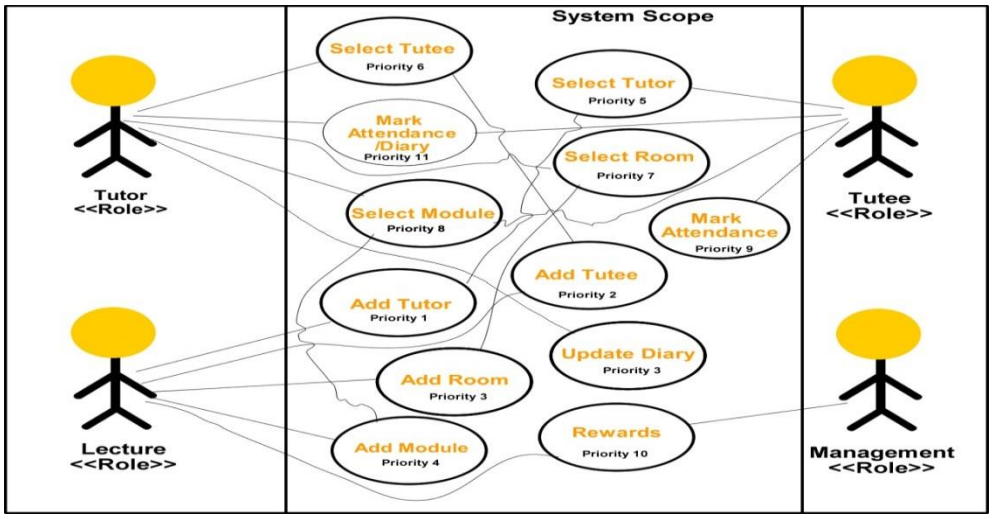


Figure 2-9: System Use Case Diagram

Phase 4: High level use cases to be developed in this phase. The smaller activities will be the names of high level use cases which are used in the transition from SSM to UML use cases. Objects belong to each use case will be named by underlined nouns. The following use cases where determined to represent the PTS. Each is represented as a tabular format.

Use case 1	Select tutor
Actor	Tutee, Lecturer
Pre-condition	Familiar with the subject area in which a tutee needs support
Description	Select a tutor that has knowledge of the module in which a tutee needs support and proceed to the next step of room or tutee selection
Exception	None
Post-condition	None

Table2- 6: PTS use case1 (Select Tutor)

Use case 2	Select tutee
Actor	Tutee, Lecturer
Pre-condition	Know the subject area which a tutor is most confident to teach
Description	Select a tutee that needs support in the subject area of interest and proceed to the next step of room selection
Exception	None
Post-condition	None

Table2- 7: PTS use case2 (Select tutee)

Use case 3	Select room
Actor	Tutor, Tutee, Lecturer
Pre-condition	Tutor or tutee or both have already been selected
Description	Select an available room which is convenient for both tutor and tutee. The room must have favourable conditions to carry out the sessions and disability access in case there is a disabled student booked into a session
Exception	None
Post-condition	None

Table2- 8: PTS use case3 (select room)

Use case 4	Schedule sessions
Actor	Tutor, Tutee, Lecturer
Pre-condition	Tutor, tutee and room have already been selected at this stage
Description	Select the date, time and duration of the sessions
Exception	None
Post-condition	A system to notify tutor or tutee of a booking made

Table2- 9: PTS use case4 (schedule session)

Use case 5	Mark attendance
Actor	Tutor, Tutee
Pre-condition	Must have attended a session
Description	Tutors and tutees to mark their attendance to sessions, so lecturers and management can use this to measure the success of the system, and lecturers and management can reward tutors
Exception	None
Post-condition	None

Table2- 10: PTS use case5 (Mark Attendance)

Use case 6	Allocate and reward tutors
Actor	Lecturer, Management
Pre-condition	Tutors must have attended the scheduled sessions
Description	Lecturers to allocate the rewards due to a tutor and management to redeem the rewards to tutors
Exception	If a tutor does not want any rewards
Post-condition	Alert a tutor that rewards have been redeemed

Table2- 11: PTS use case5 (Allocate and reward tutor)

Phase 5: Develop complicated use cases (multi-level). Breakdown the complicated use cases so that only a few high level use cases are derived from low level activities. The derived use cases were:

1-Select tutor 2-Select tutee 3-Select room 4-Schedule sessions

Table2- 12: PTS high level use cases

Phase 6: Identify top level objects. Objects are represented as classes (table2-13).

Tutee Class	Tutor class	Lecturer class
Management	Reward	Attendance
Session	Subject area	Date
Room	Module	Time
Duration	Booking	

Table2- 13: PTS top level objects

Phase 7: Map the required (top-level) services into objects, and then the objects are mapped to business processes and activities. This mapping is presented in Figure 2-10, which represents the business object model (Joseph Ucizi Mtenje (2010)).

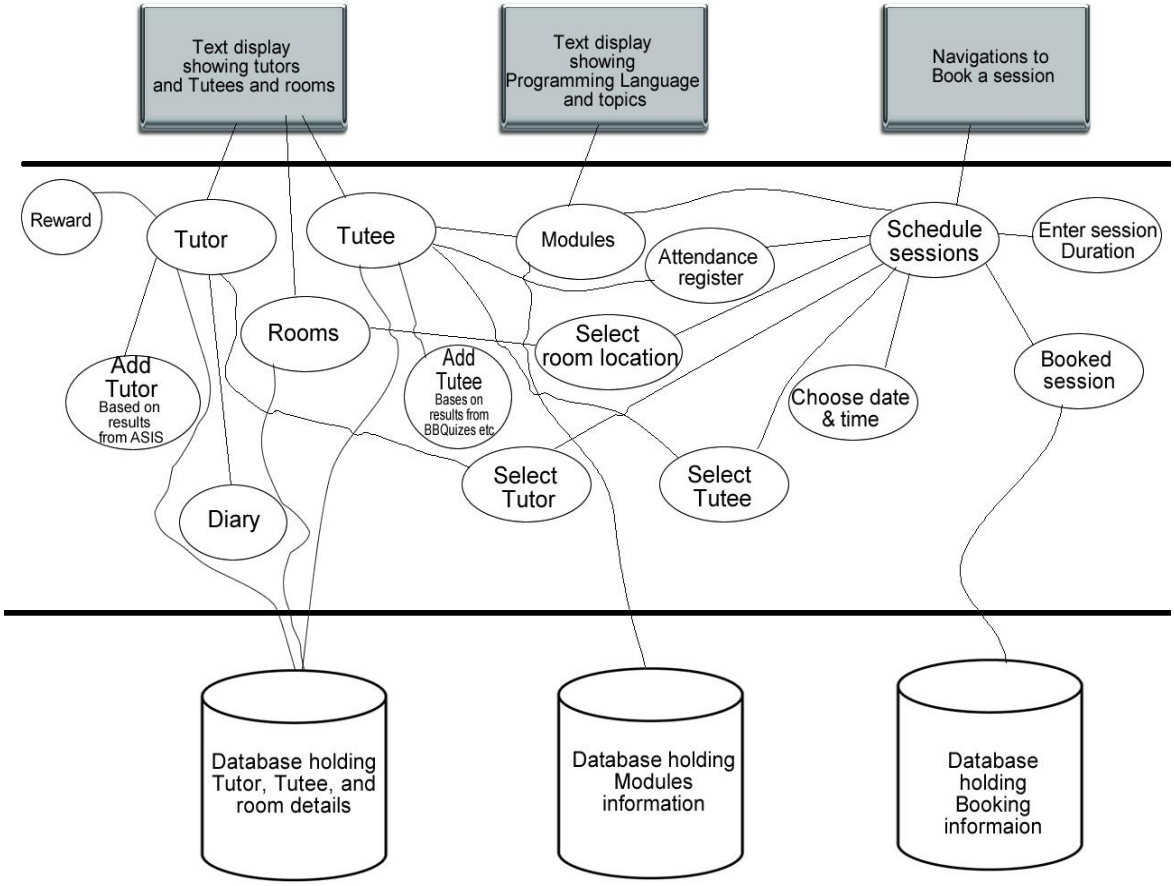


Figure 2-10: Business Object Model

Phase 8: The analysis of the UML diagrams will be continued based on the framework to improve the application design. Joseph Ucizi Mtenje (2010) cited Lane and Galvin (1999) whom were mentioned that the advantage of this transition process is that "there is no paradigm shift in the modelling language; the CM is built from activities while Use Cases describe activities. This therefore seems to be the most natural transition". Also they supported the idea of using SSM components through this transmission increase the understanding of the conversion process.

This work previous work (Salahat et al., 2009) stated that when the SSDDDF is going through the process of converting from SSM soft language to UML diagrams, it requires mapping of the activities from SSM conceptual models, after a proper understanding of the user requirements and problem situation has been gained, to use case diagrams that represent the functionality of the proposed system. While still maintaining the user requirements and business activities from the conceptual models in a one-to-one relationship, this mapping will result in some conceptual model activities being combined and some decomposed. The use case diagram is part of the use case model which represents the organisational business process, and it will be the basis for modelling the object-oriented domain model. The use case diagram provides a hierarchy of business activities concerning the stakeholder goals that led to the need to develop the system, as defined in the problem definition in the SSM stage. The conceptual models are arranged in a hierarchy whereby the more primitive and elementary business activities will be lower than the others. An chart of the conceptual model will represent the individual business activity of that part (Salahat et al., 2009).

2.6 Gaps in Knowledge in the Literature

This thesis recognized two gaps in the literature that are addressed below and has attempted to fill these gaps. The identified gaps are:

1- Eric Evans (2004) maintains that many developers who met them do not like the idea of having a common language, because the domain experts will find their concepts too abstract and may not understand the components of the model. However, he argues that "If sophisticated domain experts don't understand the model, there is probably something wrong with the model". Also, it is imperative for the stakeholders to understand the model as they are the ones specifying the requirements. Therefore, it is imperative to have a common language among the stakeholders and developers for high collaboration and coordination to avoid the IS failures. This is an important argument and this thesis has considered it in attempting to find an alternative to UL in order to fill the first gap in knowledge. This research builds on the work done in 'Domain Driven Design Framework' (Evans, 2004) but, as the author has disclosed, there is room for improvement in the 'ubiquitous language', which is considered as the *first gap.*

2-Related to this gap, understanding all system aspects ('hard' and 'soft') requires the adopted framework to handle all these aspects. However, the problem of understanding the

output of development work has already been raised by the author. This raises the argument that the adopted approach - DDD - is not able to fully address this issue, and that another enhancement is required in addition to the UL enhancement. This is considered *the second gap,* as one methodology or framework may not be enough to develop the system. Avison et al., (1990) argue that all ISDMs have limitations, and it is expected that these methodologies can be improved in the future. This thesis has considered this argument and tried to improve DDD by integrating different tools in a proposed new framework. This research introduces the new 'Soft Domain-Driven Design' approach as an extension to DDD, which adopts 'soft language' (SL) as a complement to 'ubiquitous language' in order to handle the problems explained above. The new 'interpretive ubiquitous language' is developed by the SSDDDF and, to distinguish it from the one discussed by Eric Evans, the name 'soft language' is used, which is denoted in this thesis as SL.

Part2: Literature Review: ISD selected tools

2.1 Introduction

This section explores the different information system development tools such as UML, SSM, DDD and Sogyo DDD in a comprehensive manner. The different UML models have been explored and discussed. Also, the implementation patterns of the information system development is presented and explained. These tools separated here for more descriptive and focus to be more clearer since they are selected and integrated together to propose and develop the framework SSDDD.

2.2 Unified Modelling Language (UML)

In 1997, the 'Unified Modelling Language' (UML) was introduced and established as a standard by the Objects Management Group (OMG) to allow developers to describe the structure and design of the software systems using models (OMG, 2005). UML defines a number of diagrams that can be used to describe an evolving software system; it does not, however, describe a method for actually building the software. UML is widely used as a tool in different agile methods and frameworks for modelling business processes and system functions. The next section will show the importance of using UML in different agile methods. 'Unified Modelling Language' (UML) was used as for software modelling and design to represent the ubiquity of object oriented programming through UML when comes to the design phase (Fowler, M. & Scott, 2000; Flower, M. (2004)). Various different diagrams are defined by UML, such as the use case diagram, sequence diagram, activity diagram, class diagram and others. Joseph Ucizi Mtenje (2010) cites Mishra (2004), who classifies UML into different models as represented in figure (2-12) (Mishra, 2004).

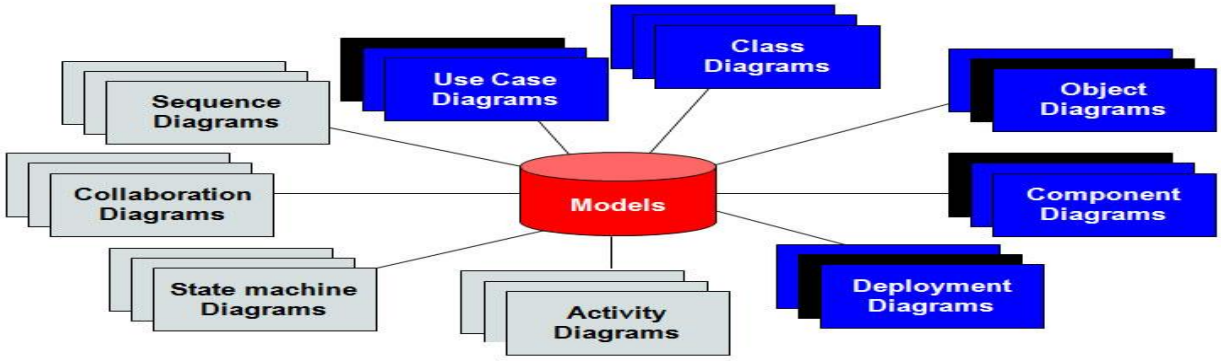


Figure2- 12: UML Models

Shoval, Yampolsky & Last, (2006) mentioned that the use case and class diagram are an important UML-based methodologies tools. Use case is widely used as an analysis tool to analyse the functional requirements, and the class diagram is used to model the problem domain. In this thesis, the proposed framework has adopted different diagrams from this model to represent different system views (layers) as explained above. These include use case diagrams, class diagrams, activity diagrams, a component diagram (replaced by Naked Objects implementation pattern) and SSM conceptual models which are mapped to use case diagrams.

2.2.1 Use case diagrams

A use case is defined by Lunn (2003, p.137-141) as a possible sequence of transactions performed by a system in a particular environment related to a particular goal to provide a measurable result for the actors. It can be represented as a diagram called a use case diagram or through a textual format called a use case proforma. A use case diagram is made up of three key elements, which are actors, use cases and the relationship between them. An actor may be a user (person or thing) of the system or another system, while a relationship is a link between actors who use 'use cases', and sometimes a 'use case' may use another use case or actor. A use case is drawn as an ellipse, and the use case description is represented in a table called a proforma which describes the behaviour of the use case. The following figure (2-13) represents the use case:

Figure2- 13: Use Case

The second element of the use case diagram is an actor. An actor is actually not a person but a role, because one person may have several roles in a system. An actor is drawn as a stick person:

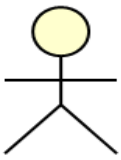


Figure2- 14: Actor

The third element of the use case diagram is the relationship, which is drawn by an arrow line as follows:

Figure2- 15: Relationship

In this case, the arrow shows that the actor uses the use case. However, there are different types of links between use cases. These links represents relationships, and there are two types of relationship:

1- Include: this means a use case *must* call another use case to perform a function (Figure 2-16)

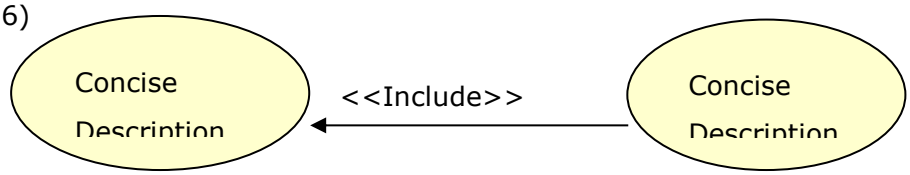


Figure2-16: Include Relationship

2- Extends: this means a use case *may* call another use case to perform a function (Figure 2-17)

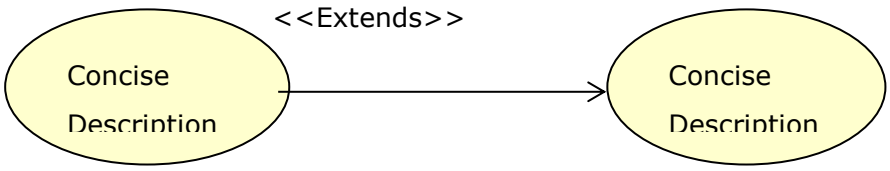


Figure2-17: Extends Relationship

The use case diagram is used by this thesis as a transition bridge from SSM conceptual model to UML use case diagrams. The following figure (2-18) represents an example of a use case diagram:

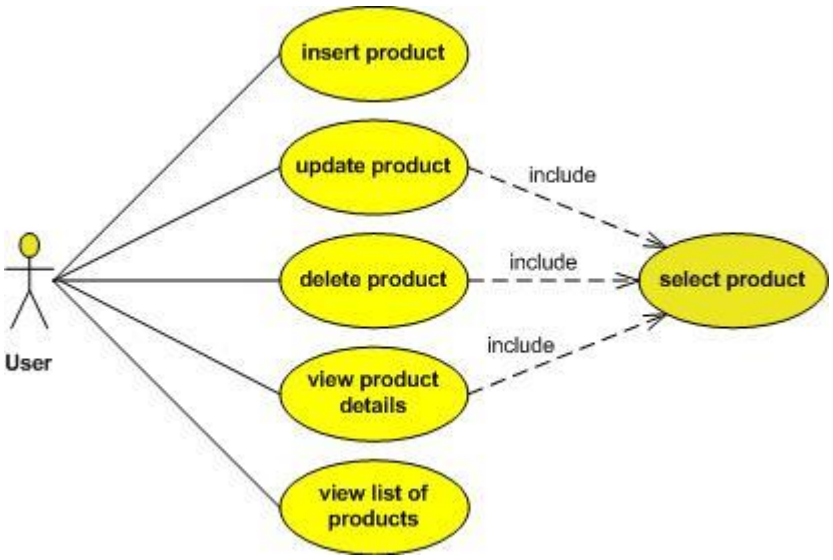


Figure 2-18: Product Management Use Cases

In order to find the use cases of any investigated domain, this thesis suggests a technique for converting from SSM conceptual model to UML use cases. This will be explained in the 'Transitioning from SSM conceptual model to UML use case' section.

2.2.2 Activity diagram

The activity diagram is defined by the UML (OMG, 2007) as a diagram to model procedural actions, the sequencing of actions and conditions for coordinating behaviours. Therefore, the activity diagram describes the dynamic features of the system. It is a flow chart diagram which represent the flow between different activities (different operation of the system). To draw the activity diagram, activities, associations, conditions and constraints must be determined first (OMG, 2007). The following figure (2-19) represents an order management system activity diagram.

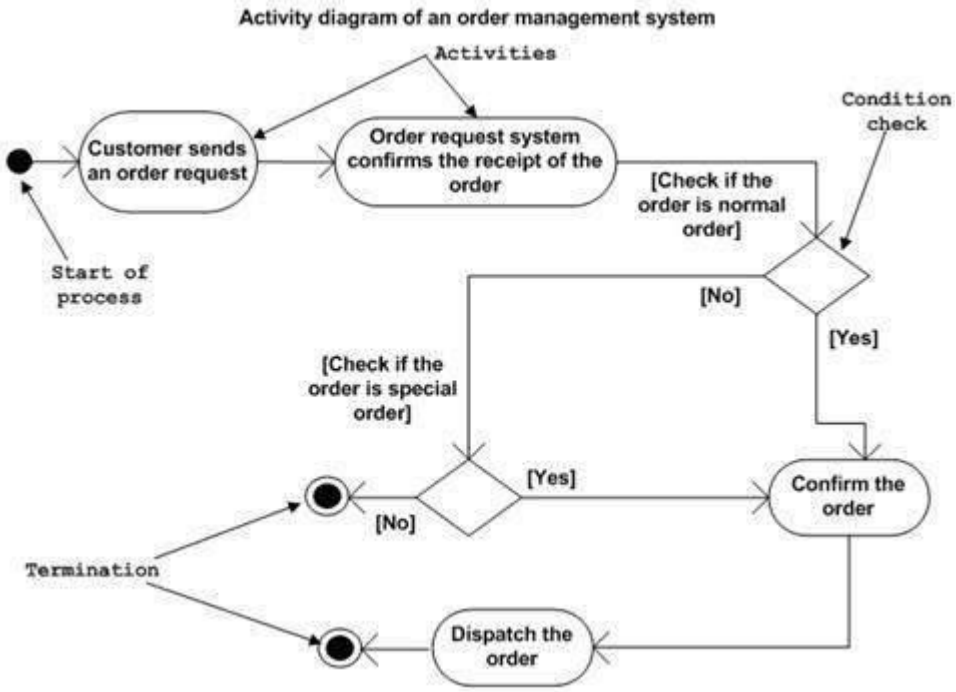


Figure 2-19: Activity Diagram of an Order Management System (Tutorialpoints-UML)

The activity diagram is one of the UML modelling tools which has been adopted by this thesis to illustrate and evaluate the SSDDD framework in Chapters 4 and 5. Different case studies are used for this purpose.

2.2.3 Class diagram

The class diagram as it was defined by (OMG, 2007) as a diagram to represent the domain model which can visualize, describe and document the system aspects, and thus construct the executable code of the software application (OMG, 2007). Class diagram consists of a group of classes and their attributes, the relations between different classes, interfaces, and constraints. Class diagram is compatible object oriented programming and it can be mapped into object oriented programming codes. The following figure (2-20) is an example of a class diagram taken from the work of students following the 'Methods and Modelling' module in 2011, which was used to evaluate the SSDDD framework as a guided learning approach for teaching ISD. The diagram represents one of the case studies used by the module - the 'Combined Studies' system.

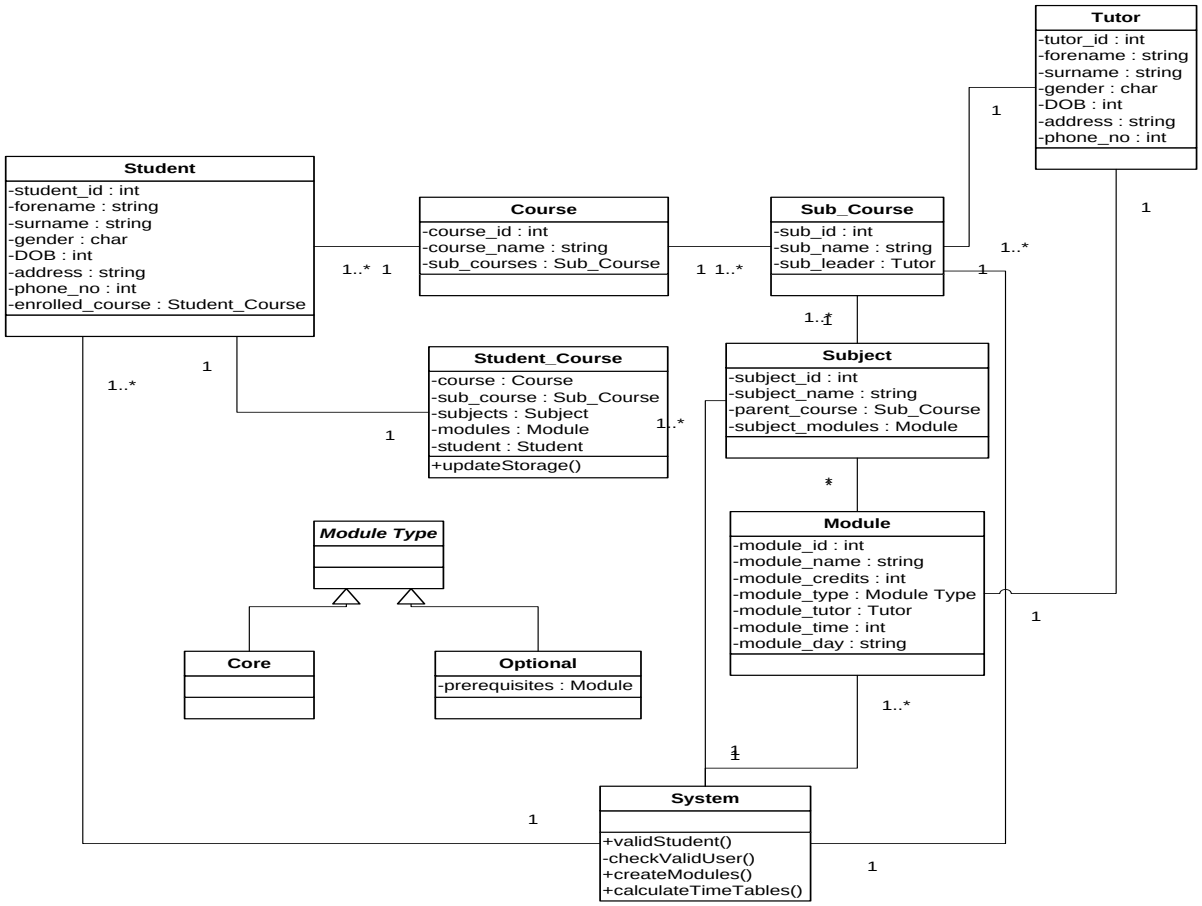


Figure 2-20: Class diagram of Combined Studies System (students' work, 2011)

2.2.4 Sequence diagrams

The sequence diagram is a popular UML artefact for dynamic modelling to identify the behaviour of the system. Sequence diagram purpose is to show and model the logic of the system being investigated. For business application development, sequence diagrams and the class diagram are the most important diagrams in the design phase. Sequence diagrams are used to model the usage scenarios of the system, the logic of methods, and the logic of services. The following figure (2-21) represents enrolling a student in a university seminar (agilemodeling.com).

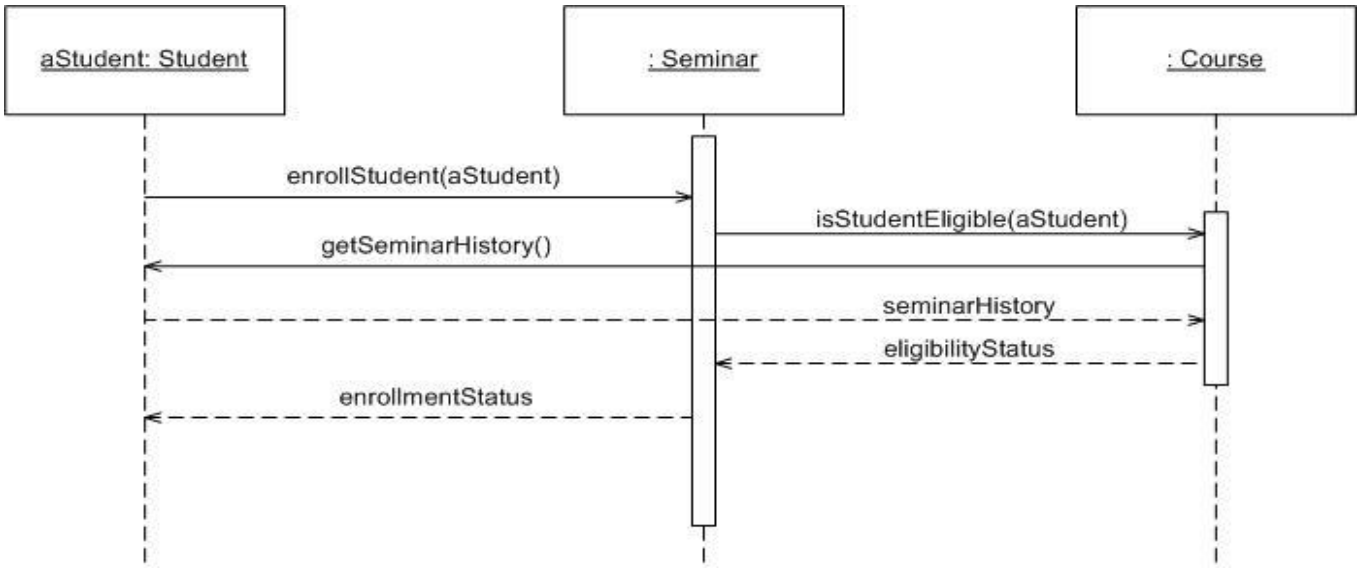


Figure2- 21: Enrolling a Student in a University Seminar

The three types of diagram reviewed and explained above have been adopted by the SSDDD framework as UML diagrams in addition to SSM diagrams. This thesis investigates the combination of UML and SSM diagrams and the application of these to different case studies. The transition from SSM to UML use case diagrams is reviewed and discussed in this chapter, and a discussion of the application of this approach is available in other chapters.

2.3 Soft Systems Methodology

Checkland, 1981 and other researchers developed a methodology called Soft systems methodology (SSM) at Lancaster University. SSM is based on system theory which request to decompose the system into small components in order to study and understand them.

Systems theory is a holistic approach since its concentrate on studying the whole picture of the system by exploring the relations between different components of the system under investigation. SSM is not an ISD methodology; it is a problem solving methodology which was used to investigate problems from different domains such as environmental sciences, biology, and systems analysis. Different researchers adopted SSM for different applications, such as the work of Brian Wilson, 1990 at Lancaster University who was used the methodology to analyse the business information systems. Also another attempt done by Avison's, 1990 who incorporated it into systems design work through the methodology

'Multiview2'. Others have made efforts to incorporate SSM with UML (Bustrad, et al, 1999; Steve W. & Judith Hopkins, 2002; Al Humaidan, 2006; Sewchurran & Petkov, 2007)).

SSM was declared as a methodology to understand and structure the complex messy situation, by constructing conceptual models of the human activity system (HAS) them compare them to the real world system. Conceptual models were considered as a potential real world systems, but not a real representations of the real world system.

So, SSM is a methodology to structure thinking about the system but not to analyse it, and it is useful since it allow the involvement of different stakeholders whom interesting about the solution of the investigated business domain problem.

Checkland's seven stage methodology is represents in Figure 2-22.

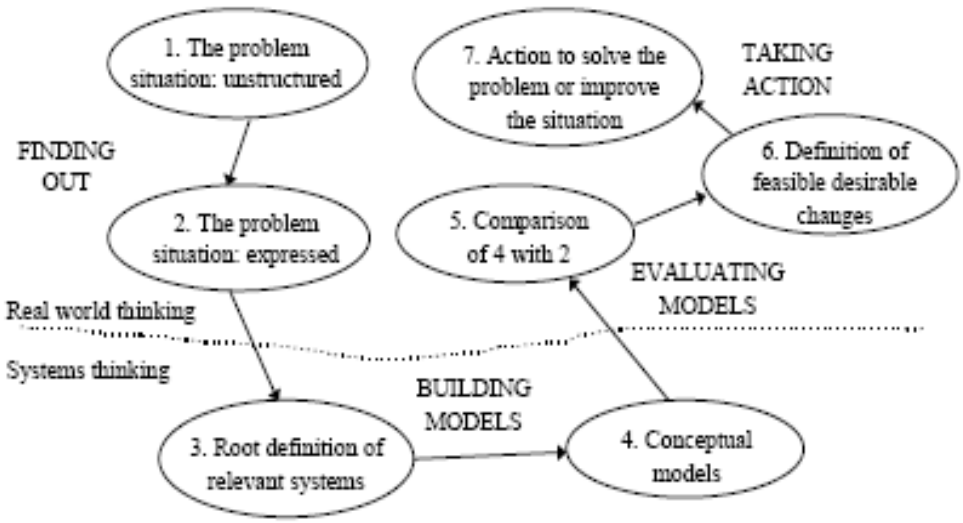


Figure2-22: Checkland's Seven-Stage Soft Systems Methodology

2.3.1 SSM and information systems

SSM was declared as a methodology for problem solving but it was used in Information System domain specially information system management, strategic information system, and business analysis. SSM is not for computer Information system design, but to understand how to think about the problems available in the domain to be computerised.

An attempt by Brian Wilson (1990) has been done to model the different stakeholder view (W's) to handle all activities of the business domain. This attempt is considered an extension to SSM and the 'consensus primary task model' CPTM was developed to represent the majority of activities agreed by all stakeholders. This can be constructed by combining the same activities available different conceptual models (different stakeholders a greed about them) in order to represent the business domain problematic situation. These activities will be examined and if any is a larger one it will be decomposed to smaller activities. Input to carry each activity will be determine and output also in order to formulate 'information categories'. This will make the information requirements clearer and complete without any duplication and shortness.

2.3.2 SSM strengths and weaknesses

The strengths and weaknesses of SSM are linked to two important issue (Paul Lewis, 1995):

- First issue is relating to its ability to handle the complex situation facing people during the analysis stage; this is good to build the system but may be cause an ambiguity to the system developer.

- Second issue, it's a logical methodology starting with investigating the problem of the business domain, then proceeding to conceptual models development.

Some researchers like Kingston (1995) argues that a lot of inputs and outputs available in the SSM models without identifying which output belongs to which input. So it requires to improve the whole system in order to get any specific improvement. This will make it difficult to develop and implement the soft system model.

This research adopted soft systems methodology to enable investigation of the different projects used, such as the 'Peer-Tutoring System', to a greater depth in the sense that the models in SSM will help to build up a debate which will enable an understanding of the requirements of the systems to be developed. It will also help to prepare a use case models that will aid application development (Checkland, 1989). Using SSM, different stackholders views can be expressed and this will help to solve the problem through learning rather than adopting a new solution (Davies & Ledington, 1988, cited by Winklhofer, 2002). Therefore, the application of SSM to business domain modelling supports project development by demonstrating requirements more clearly and enabling a better understanding of the whole business domain and functional system. The resultant software system will be more helpful

to users, as it will meet their needs. It also gives the project a good likelihood to be accepted by different stakeholders

2.3.3 SSM rich picture

Rich picture is a key tool of SSM and is a graphical representation of the whole situation. Anything can be used in this picture to make the problematic situation clearer.

Developing SSM rich picture required the analyst to be sure that the perception corresponds with each stakeholder, he understand the situation, and identify other related issues of the domain like ethical issues and disagreements (Kingston, 1995). According to Checkland (1990) a rich picture represent a way of asking stakeholders the question "Have we got it right from your perspective?" in order to be sure that the work in the project is in the right direction. Rich picture allows the investigator to develop a holistic view about the problem situation. Figure 2-23 presents an example of a rich picture about classroom interaction (lecture situation) (Patel, 1995).

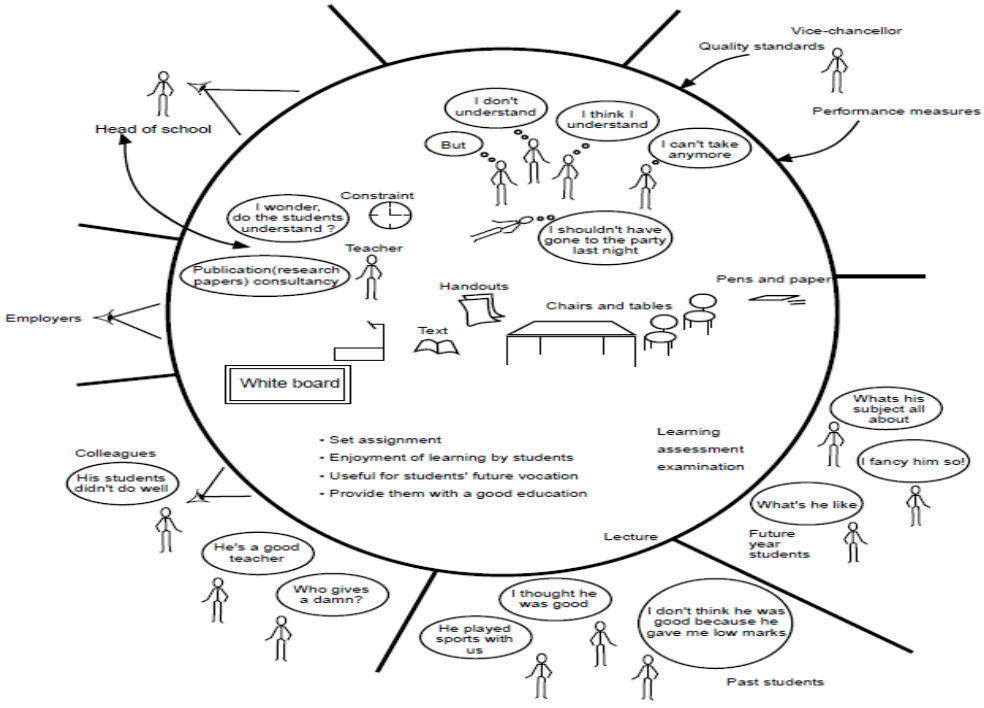


Figure2- 23: Rich Picture of Classroom Interaction

2.3.4 Root definitions

Root definition (RD) may be described as: "a short textual definition of the aims and means of the system to be modelled" (Rose, 2002). The main purpose of using 'Root definition' is to determine the purpose of the system and the interested parties. RD is constructed from the different views of these parties based on their expectations about the system functions. In other words, root definition can be used to represent the mission of the system and look at the problem situation from different points of view. Modelling a system using root definition has been described as a movement from the real world to systems thinking about the real world (Checkland & Scholes, 1990). Williams (2005) mentions that during the root definition stage, points of view from the different stakeholders are drawn out from the rich picture and presented within a structured development process. The following examples illustrate two root definitions derived from the rich picture presented in Figure 2-24, taken from Patel (1995), which represents the lecture situation (classroom interaction):

Root Definition 1

"A lecturer owned system, jointly operated by the lecturer and students with the available teaching and learning resources to ensure that students pass or achieve higher marks in assignments and examinations to a realistic ceiling, being the joint aim of the lecturer and students, subject to the requirements and constraints of the University of Luton."

Root Definition 2

"A lecturer and students' jointly owned and operated system with the available teaching and learning resources to ensure that students learn relevant knowledge which is worth learning for their vocational and educational benefit while enjoying the process of learning by delivering lectures subject to the various constraints of time, learning and absorption rates of students, limitations of the room and other teaching and learning resources, meeting both the required quality standards expected by the University of Luton and the lecturer's performance measurement criteria relevant to the system."

Root definition 2 will be used as an example to extract the conceptual model from as it represented in the next section.

2.2.3.5 SSM conceptual models

Conceptual modelling process represent a step away from the real world modelling and concentrate on abstractions. A conceptual model is an abstract representation of concepts (entities) and terms, and the relationships between them. The purpose of a conceptual model is to convey the meanings of the concepts and terms used by the domain experts and to find the exact relationship between these concepts. The conceptual model is extracted from the root definition. The conceptual model represents the human activities system (HAS).These conceptual models will be the bases from which to link SSM and UML through use cases. The next section will review the linking process. Figure 2-13 represents the conceptual model of teaching and learning (Patel, 1995) which was derived from Root Definition 2 mentioned above.

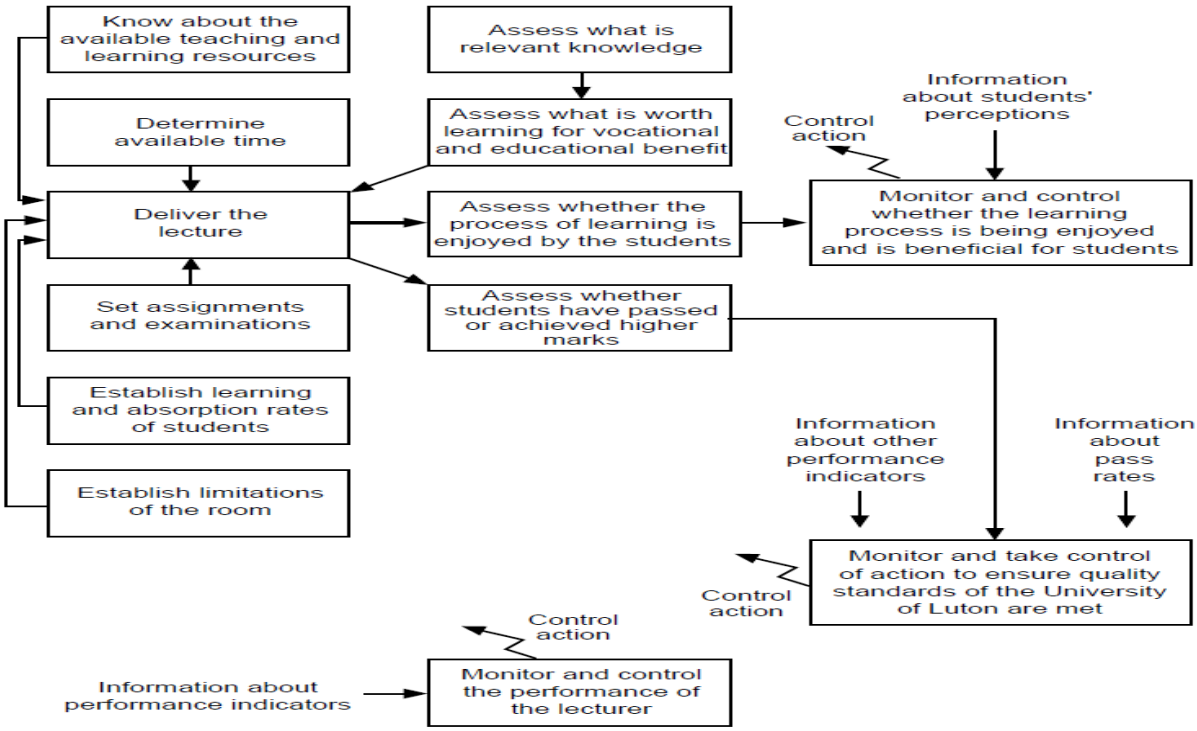


Figure 2-24: Conceptual Model of Teaching and Learning

2.4 Domain-Driven Design

Domain-driven design is concerned with mapping the business domain into software artefacts that can be used to develop the final software system. The following sections will explore this idea and show how it can be related to business domain modelling and implementation.

2.2.4.1 The Software Development Process

The application development process consists of a group of phases and elements to be followed for developing a software system, and these vary depending on the methodology or development approach used. There are many approaches for software development and among these, DDD (Evans, 2004) was introduced to manage the complexity of the application development process. Michiel Uithol (2009) presents the application development process in the context of DDD as in Figure 2-25. Understanding these stages is a major prerequisite to exploring the nature of DDD in detail. The problem domain at the top of the model represents the basic idea about the final achievement of the developed application. This will be refined and a requirements specification document will be produced to be used in the design phase. The design phase will transform the requirements specifications into an 'application model', and the requirements will be refined and adjusted during this phase to fit with the application model. This will be followed by the preparation of the 'application model specifications' before starting the implementation phase. An implementation corresponding to the application model will be produced during the implementation phase, followed by the structuring of codes to reflect the behaviour of the implemented software system.

In the development process, transformation from the problem domain into an application model leads to a 'domain model'. Similar to the previous stages, any refinement in the application implementation will refine the application model.

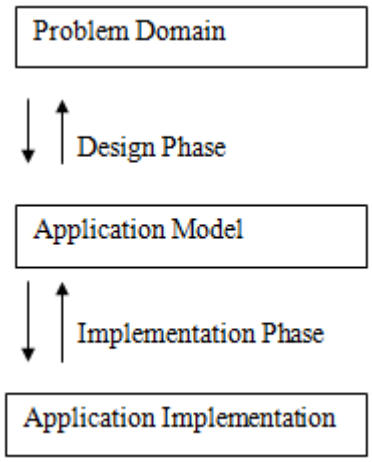


Figure2- 25: The Development Process

2.2.4.2 DDD Philosophy

A software development project aims to solve a specific problem for a given domain by developing a successful software system to support the business activities of the domain and run them successfully. The main philosophy of DDD is that "the focus must be on the domain and its logic (i.e. the business logic) in any software development project" (Uithol, 2009). This is an important concept, since the activities embedded in any domain reflect the real business artefacts which must be considered, rather than the technology.

Domain-driven design is not a development method, but it is oriented toward agile development methodologies and utilizes well-established software design patterns (Hoffmann, 2009). It is an approach which tries to handle the complexity of software development by mapping business domain concepts into software artefacts to create better software by focusing on the domain model and the logic embedded in it (business logic) rather than the technology. Other methodologies focus more on the technology, through the software development process, and because of that the resultant models do not reflect the domain business logic as it is understood by business experts (Evans, 2004). The complexity of the software development lies within the problem domain, and the separation of the 'application model' and implementation keeps the focus on this problem domain, i.e. domain logic or business logic (Evan, 2004; DDDC, 2008; van Dillen, 2007). In the development process, the design phase involves developers and domain experts who collaborate to produce the application model. Jacopo Romei (2009) summarises the three words represented by DDD by suggesting that 'domain' is what inspires our solutions, 'driven' is where we find our solutions and 'design' is what provides us with solutions. This view presents DDD as a way of coping with problematic situations and helping developers to be good designers. Jak Charlton-thinkddd.com, (2010) describes DDD as an architectural methodology for evolving a software system that is closely aligned to business requirements. However, DDD is not focused on how but on what and why, and it is not always the easiest, or even the best, solution to follow.

This thesis considers this argument and seeks to find a way to make DDD an easier and better solution in most cases. The core concept of DDD is the development of a 'ubiquitous language' (UL) as a means of communication between business domain experts and software developers; this is intended to guide and support the extraction of the domain model which reflects the business activities embedded in the organizational business process. This model will be used as a communication guide through the remaining stages of the software system development process. The following subsections explore both the

ubiquitous language and the domain model in more detail, in order to show how they are connected, how the language is used through the development process, and the nature of different types of domain model.

2.2.4.3 Ubiquitous Language

2.2.4.4 The nature and the role of the language

If an ideal software development environment is available, domain experts and developers must sit together in order to discuss different issues related to the development of the new software. Domain experts have limited understanding of the technical concepts of software development, and software developers have a technical view of the system which does not reflect the domain experts' understanding and requirements. Developers always use abstraction to support their design and these abstractions are always not understood by the domain experts. Here there is a linguistic divide, because domain experts describe their requirements vaguely and developers struggle to understand a domain which is new to them. Without using a common language to communicate, the developers start translating to domain experts and domain experts translate to developers and sometimes developers or domain experts start translating to themselves. This will lead to misunderstanding and produce inconsistent materials which will affect the development of the domain model negatively, so that the software which is finally implemented will not reflect the real business activities. There is therefore a need for a common language to control such communication and to help in producing a robust model which can be a backbone for this language. That language can function as a ubiquitous language in the team's work (Evan, 2004).

Ubiquitous language, therefore, is a communication language between the different system stakeholders. It helps the software developers, business experts and others to use a common communication language in writing, diagramming and speech. Ubiquitous language is designed to ensure that all the team members communicate in an appropriate way and understand each other. It will help the team to create an understandable application model. It has been mentioned that the major reason for software system failures is related to poor business process modelling, which results in production of a poor domain model. According to Jak Charlton-thinkddd.com (2010), a poor domain model can be produced if the problem of communication between the team members is not resolved, thus leading to misunderstanding and an inconsistent model. Furthermore, business process modelling must consider all organizational business process aspects, both 'soft' and 'hard' (Salahat et al., 2009; Al Humaidan, 2006); this comprehensive view will help to model the business

processes in a proper way and lead to a proper domain model. To achieve this, the ubiquitous language must be improved by adding all the 'soft' artefacts related to the business processes. This suggests the addition of texts and diagrams as a result of using a 'soft business process modelling approach'. Soft system methodology (Checkland, 1999) is a well-known methodology and is proposed for use here as a soft business process modelling approach. However, this thesis also suggests the use of an alternative language as a complement to UL, a 'soft language' which may offer an improvement to the issue of UL; this will be discussed in the 'Alternative to UL' section.

2.2.4.5 The vocabulary and usage of the language

The vocabulary of ubiquitous language includes the names of classes and operations. It includes terms used to discuss the exact rules of the model, supplemented with terms from high-level organizations like 'context maps' (Evans, 2004). It also includes the names of patterns used by the team and applied to the domain model. This is a model-based language and is used to describe the artefacts of the system, tasks and functionality. Using the language in the context of implementation will help the developers to point out key issues, which will encourage the domain experts to find alternative solutions. Using the language and raising comments when not satisfied will ultimately lead to a complete model through different iteration steps, and this model will combine simple elements expressing complex ideas. However, based on the argument proposed above, UL can be improved further if 'soft' artefacts generated by SSM are added to the language; this would enable all organizational business process issues to be considered in order to develop a comprehensive domain model which can be used in the implementation of the software system.

2.2.4.6 Alternative to UL

Eric Evans (2004) maintains that many developers who met them do not like the idea of having a common language, because the domain experts will find their concepts too abstract and may not understand the components of the model. However, he argues that "If sophisticated domain experts don't understand the model, there is probably something wrong with the model". This is an important argument and this thesis has considered it in attempting to find an alternative to UL.

The domain model is extracted based on the developed ubiquitous language, which supports the incorporation of all business activities of any given domain into the domain model; otherwise the extracted domain model will be inconsistent and incomplete. The process of

extracting the domain model depends on business logic, but the tools used for modelling the business logic may not be understood by the domain experts. The development of a ubiquitous language is designed to enable a common understanding between business experts and software specialists, and to allow people from all backgrounds to understand the tools and concepts required for mapping the business activities into a domain model. Nevertheless, it may happen that some or many of the business experts do not have the required technical background to apply and develop the concepts of UL as a communication tool, and this could lead to problems in the development of the domain model. The problem boils down to the difference between an objectivist approach (e.g., as in class diagram modelling) and an interpretive approach such as that adopted in the social sciences (e.g., as in structuration theory Giddens, A. (1984)). Therefore, it could be argued that interpretive approaches could help in the difficult task of developing a ubiquitous language, and soft system methodology might help here. SSM is firmly rooted in an interpretive mind set, as has already been introduced and explained in the previous section. Recently, other authors (Wang, Q., Chen, J., Wen, H., Liu, L., Lian, J., Bai, M., ... & Pei, Z. ,2014) suggested a Domain-Specific Language (DSL) as a standard communication tool between the team members, which aim to address similar problem to what done and solved by this research. They didn't use SSM but they tried to be similar to DDD. Chapter 4 will introduce the new 'Soft Domain-Driven Design' approach as an extension to DDD, which adopts 'soft language' (SL) as a complement to ubiquitous language in order to handle the problems explained above. The new 'interpretive ubiquitous language' is developed by the SSDDDF and, to distinguish it from the one discussed by (Eric Evans, 2004) the name 'soft language' is used, which is denoted in this thesis as SL.

2.2.4.7 The nature of the Domain model

The domain model represents deep knowledge since it reflects the different views of all project stakeholders. It is an abstraction of domain knowledge organized in a proper way and as such, it is distilled knowledge and a backbone of the language spoken by all stakeholders (the project team members). Stakeholders often have different views of the model, and this requires intensive collaboration between domain experts and software developers to create and maintain the model through a 'knowledge crunching' approach. 'Knowledge crunching' (Evan, 2004) is an extensive exploration of the domain and a continuous learning experience. It can be achieved through brainstorming, talking, experimenting, sketching and diagramming knowledge from domain experts, experiences from current and legacy systems, etc. It is very important to distil knowledge as much as

possible to enrich the domain model and to utilize this knowledge in the later stages of software development. Therefore, the model is a result of communication between different team members, including business experts, software developers, users and others. From the technical point of view, the domain model consists of 'domain-related functionality' and 'domain-independent functionality' (Uithol, 2009). It comprises a group of services to facilitate the usability of the domain model. The application implementation includes a separation between domain-independent (service implementation) and domain-related functionality (domain implementation). Software design must be driven from this model and thus the model may be considered a model-driven design. Developing a complete and accurate domain model will help to reduce the complexity of the application model. To be an accurate domain model, all team members must be satisfied with the functions incorporated in it (this will include all soft issues related to the team and their work) and to be complete, all functions related to the domain must be presented in it.

2.2.4.8 Benefits and characteristics of domain model

The domain model helps to improve the usability and testability of business domain objects, helps the team to communicate correctly while they are dealing with the business requirements, data entities and process model (Penchikala, 2008), and is easily maintainable since it reflects the business model. To be a correct and complete model, it must satisfy a set of criteria which includes the following issues as summarised by Srini Penchikala (2008). It should focus on a business domain; be isolated from other domains in the business and other layers in the architecture; be reusable to avoid duplication in modelling and implementation; be loosely coupled with other layers in the application; and be abstract and independent of persistent implementation details.

In order to achieve the organizational goals, especially better return on investments in software development, business units and IT managements must consider a reasonable investment in business domain modelling and implementation (Penchikala, 2008). This requires investing in a good team which can demonstrate good business process modelling skills; good design and implementation skills; experience in object-oriented design and programming and soft system methodology; and communications skills. A further requirement here is the ability to develop 'soft language' (SL) as a complement to the UL developed by Eric Evan (2004).

2.2.4.9 DDD layered model architecture

Eric Evan (2004) proposed the architecture layers illustrated in Figure 2-26. This structure, called layered model architecture, aims to concentrate code of the domain model in one layer (domain layer) and to be separated from other layers (the user interface, application and infrastructure). It would be difficult to manage or maintain the code related to the business domain if it were scattered into the user interface, infrastructure and application layers. If any business rule were changed, this would require changing the code in different layers; this assumption supports the domain-driven design approach of separating the codes related to the domain into the domain layer. DDD focuses on the domain layer, and the components interact with other components in the same or other layers as depicted by the arrow directions in Figure 2-26. Each layer is specialized to manage different aspects of the software codes. The model layers and their functions, as presented by Eric Evan (2004), are as follows:

1 - *User interface layer* (also called the presentation layer): responsible for interpreting the user's commands and showing the information to him.

2 - *Application layer*: responsible for coordinating application activities, such as navigation between user interface screens and application layers and validation of user input data before passing it down to other layers of other systems. This layer does not contain any business rules or knowledge related to the domain, so it is kept thin; it does not have a state to reflect the business situation and rules, but it can have a state that reflects the progress of a program or a task for a user.

3- *Domain layer*: this layer is the heart of the business software and contains the concepts of the business domain, business rules and use cases, the state and behaviour of business entities and information about the business situation (Penchikala, 2008). It can manage the state of the business situation and contains services which encapsulate the business domain behaviour but which are not part of the domain itself.

4- *Infrastructure Layer*: this includes the generic technical capabilities to support the other layers. It supports the pattern of interactions between the four layers through an architectural framework. It provides communications between different layers and acts as a supporting library to other layers.

Some authors support this layered architecture (Evan, 2004; Penchikala, 2008; Wang, Q., et al.,2014), but other authors argue about the direction of interactions from up to down,

which prevents interactions between layers from the lower level to those in the upper level as a refinement process (van Dillen et al., 2007). They suggest another layered structure called 'Sogyo' which is discussed in the next subsection. The remainder of this section deals with the different authors who support the layer architecture.

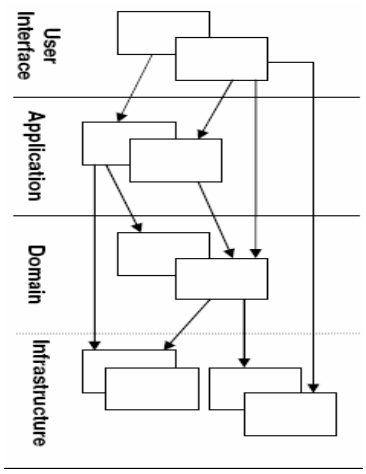


Figure2- 26: Common Layered O-O System

Domain-driven design focuses on modelling the business domain to include the different artefacts required to map it into a software support system. As stated by Srini Penchikala (2008), based on the domain-driven design approach, a domain modelling and implementation project includes the following steps which were presented in table 2-14:

1- model the business processes with business experts	5- design the system using the domain
2- document the modelled processes into the ubiquitous language	6- conduct software testing
3- find all services related to the modelled processes	7- refine and refactor the domain model based on the design and implementation
4- find and document the state and behaviour of all objects used by services required by the modelled processes	8- repeat the above steps using the updated domain model.

Table2- 14: Domain modelling and Implementation project steps

The above perspective of modelling and implementing a business domain is similar to other approaches to software development. The difference here is that there is more concentration on business domain modelling, which is the main contribution of DDD. This view indicates that DDD begins after the domain modelling ends. This supports the proposal of this thesis, which is to add a 'soft' perspective to business domain modelling before starting the DDD approach. The proposed SSDDD framework (Salahat et al., 2009) is based on Srini Penchikala's (2008) approach, but may be summarised in the following steps:

1- model the business domain using all business and technical artefacts and generate UL (add a new perspective (soft perspective depicted by SL) to this approach to handle the soft issues of the organizational business process during the modelling phase)	4 develop the software system based on the modelled domain and the generated UL
2- find services, states and behaviour related to the modelled processes	5- refine the developed software developed using the domain model based on SSM learning and exit
3- design the system based on what has been done in the previous steps	

Table2- 15: SSDDDF proposed steps

This may lead to an improvement of the DDD approach. Based on the above procedures of business domain modelling and implementation, all perspectives of the organizational business process will be modelled and used to develop the software system. A comparison between DDD and SDDD was presented in Chapter 7. However, Ramnivas Laddad (2009) suggests different steps for implementing a domain objects model, in which he focuses more on domain objects than services in the model. His approach consists of the following steps which they were presented in table 2-16

1- start with domain entities and domain logic	4- design the system
2- add services when the logic does not belong in any domain entity or value object (start without a service layer)	5- conduct automated tests and refactor to align the implementation to the model.
3- use ubiquitous language	

Table2-16: The steps of implementing domain objects

2.2.4.10 Building Blocks of the Model

DDD determines a set of conceptual objects to be used in the code in order to implement the domain model. Model-driven design components are the building blocks of domain-driven design, as presented in Figure 2-27 (Hoffmann, 2009) which is developed based on the work of Eric Evans (2004).

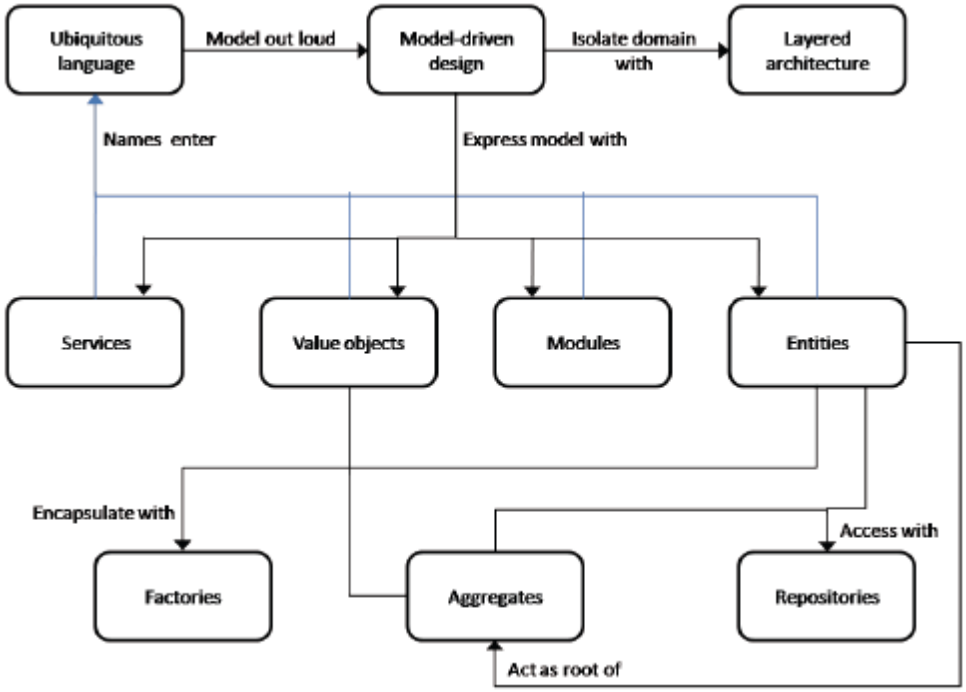


Figure 2-27: The Building Blocks of DDD

As shown in the building blocks diagram (Figure 2-27), DDD uses the architecture layer approach, ubiquitous language and model-driven design to extract the domain model from the business domain. Ubiquitous language is used to extract the model, and model-driven design is used to express the model as services, value objects, modules and entities. These names of these objects will be stored back in the ubiquitous language to facilitate communication in forthcoming stages. Layered architecture is used to isolate the business domain from other services to facilitate programming, maintenance and any other technical issues.

2.5 Sogyo domain-driven design

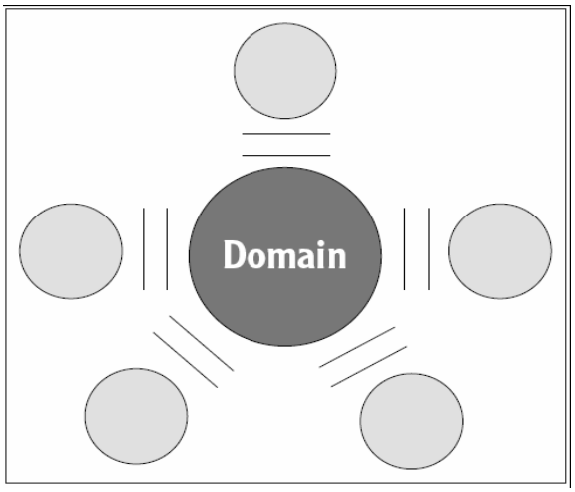


Figure 2-28: Sogyo DDD Application Model

The Sogyo DDD model uses a 'sunflower' model (Figure 2-28) (van Dillen et al., 2007), in which the domain functions are centred in the middle and services outside. In this structure, the implementation of the domain model is unaware of the services in the structure layers. The difference between the Sogyo structure and Eric Evan's (2004) DDD is that the services are not presented in one layer but in separate entities around the domain model. Also, the domain model is unaware of the elements in the infrastructure layer (van Dillen et al., 2007).

The main output of the design is the domain model. The domain implementation is independent and can operate in isolation. The double lines between the domain model and services represent a 'glue' layer which is equivalent to the application layer in Eric Evan's (2004) approach. The function of this layer is to translate actions in order to use the domain classes.

2.6 Implementation Patterns

SSDDDF suggests the use of Naked Objects or TrueView as implementation patterns. Pawson (2002) defines the Naked Objects framework as "A set of Java classes that can be instantiated or sub-classed by an application". Most business systems today have adopted the architectural pattern of having four generic logical layers, with new business concepts having been implemented in all four layers in different forms (Pawson, 2004). The four

layers as described by Brown (1995), are presentation layer, controller layer, domain objects layer, and data management layer. Pawson ,2004) mentioned that the method of layers was used before that, he argues that relationships must be available between these layers but it is a complex mapping. This architectural layers model became a generic through the years and any of each layers can be ab objected oriented behaving.

Naked Objects used an object-oriented user interface to allow the user to see and manipulate the domain objects' behaviours for any action. Pawson, 2004 mentions that domain object was represented as a user icons and all transactions required will be as options from these icons.

TrueView software is produced by by Evolving Software company registered in England and Wales in 2006. Using TrueView, the application software is created based on .NET entities(the classes developed in the UML stages). TrueView implementation pattern is used to explore the business domains and to create rapid prototypes based on domain-driven design approach, and the applications produced reflects the domain models. The company mentioned that TrueView helps to keep business logic clean, concise and focused by having an object-relational mapping facility for data persistence. The company also mentions that the application was designed to suit problem solvers, which is why it is being used in this project. It allows freedom and flexibility in DDD implementation as the interfaces can be customised, security capabilities can be added and it offers data persistence to an application. As TrueView's behaviours are controlled through attributes, it creates entity classes and relationships between them that help to keep the whole system working and deliver efficiency in the system".

In addition to the above patterns, the SSDDD framework is compatible with other implementation approaches if developers so choose. In Chapter 5, regarding the 'School Liaison Coordination System', the developer preferred to go for another implementation approach that he had mastered well before, and the system was implemented smoothly without any problems.

Chapter 3: Research Methodology

3.1 Introduction

There are different definitions of research and among of these a research is a scientific and methodical search of a data about a specific problem under investigation. A research methodology is referred as the blue print of a research, where the methods to conduct a particular investigation for the purposes of resolving an issue are explained and justified. It can be understood as the science of examining the process of conducting an investigation. Under a methodology, one evaluates the phases that are deployed by the researchers to reach specific outcomes. Also, the rationale to choose the particular methods for a specific analysis is also explained under methodology.

This chapter, therefore, presents the research methods appointed in the current study to answer the mentioned research questions. The entire investigation depends on the research methodology and it is imperative to deploy research methods for acquiring the final interpretation of the research. This chapter comprises of research paradigm, research approach, research design, data collection and analysis, validity of results and ethical considerations.

3.2 Research Paradigm

The purpose of a research is to discover and construct several ideas for the perspective of resolving an issue. It is an examination that attempts to gain knowledge, analyze issues at hand and solve it by acquiring insights into the depth of problems (Jupp, 2006). A research paradigm enlightens the general methodology of the research (Johnson and Christensen, 2010). There are two paradigms in the broad spectrum of research namely, positivism and interpretivism. Positivism is a structured method that comprises of logical deductions backed by observations. By considering and using this paradigm, the researchers will adopt a large social sample to collect general information instead of focusing details of research, and it's depends on raw numbers and numerical data (Creswell, 2013). Interpretivism is the research philosophy, which is subjective where researchers interested to highlight the research problem through presenting different facts and figures about it (Creswell, 2013).

3.2.1 Research Paradigm Adopted

This research utilizes the paradigm of interpretivism as to answer the research questions, it is imperative to gather engaging information and induce theories from that information. This investigation unequivocally trusts on a few angles, for example, individual support, notion, and feelings of the members, which required the vitality for uncovering the data. As the present research aims at investigating and implementing a multimethodology framework that addresses hard and soft requirements, qualitative and interpretive research is deemed fit to evaluate the proposed framework. For this purposes, action research is deployed where the idea is to develop the theoretical and make it practical, whilst simultaneously taking the practical and making it theoretical. The theoretical part includes the development of a new framework ('Systemic Soft Domain-Driven Design Framework' (SSDDDF)) combining soft system methodology, unified modelling language (UML) and the Naked Objects implementation pattern in the context of domain-driven design (Salahat et al.,2009). The practical part refers to the evaluation of the framework using different real world case studies from the researcher's university and through using the framework for teaching the module 'Methods and Modelling' for postgraduate students, and followed by a comparison of the proposed framework with others reviewed in the literature chapter.

3.3 Research Approach

Research approach defines the method with which a particular investigation is carried out. It describes the philosophy that drives the direction of the investigation (Morgan, G. A., et al, 2006). Quantitative and qualitative are the two research approaches that are most extensively deployed in practice (Thomas, 2003).

A qualitative approach is subjective in nature that utilizes a phenomena or setting to understand, illustrate and generate hypothesis. In other terms, such approach is adopted by those, who aim at studying their equipment in their surroundings and evaluating the phenomena through the opinions brought to them be people (Burman, 1997).

A quantitative approach is descriptive in nature and aims at reaching conclusion through facts and numerical data. It is used by the researchers to comprehend dissimilar promotional inputs possessions on the client thus enabling the marketers to find the performance of consumer (Bryman, 1984).

3.3.1 Research Approach Adopted

The qualitative approach is described by Gupta & Gupta (2011) as formal and dynamic to approach to utilize formal and informal instruments for collected data . It comprises of thoughtful inspection of the subjective information acquired from human experiences to identify the meaning behind them and analyse the information (Brace et al 2006). As the present research implements a framework for information systems development, which is adopted as a teaching approach for the learning of DDD, SSM, and UML. To estimate the benefits of the proposed framework, it is imperative to gain the feedback of the ISD developers and stakeholders, and therefore, engaging information can be extracted through the means of qualitative data. Addition to that, the feedback of learners gained through the qualitative and quantitative data.

3.4 Research Method Selection

This thesis adopted action research method incorporating qualitative methods in order to gather the data. Case studies and interviews are triangulated and used as a surveying qualitative research methods. The use of a case study approach in information systems research has been addressed and supported by Gummesson (2000) while Avison (1990) and Wood-Harper (1985) also justify the use of action research for information systems research. Gummesson (2000) mentions that it is difficult for researchers to gain reasonable access to a company to investigate and develop a detailed case study. This is where action research is of great benefit, as it supports the selection of case studies from the researcher's own work environment in order to gain easy access through the investigation phases. Therefore, this research utilizes educational environment to evaluate the efficiency of the proposed framework. This is further described in the following sections.

3.4.1 Action Research

Action research is one major and important type of research methods which defined and explored by different researchers as follow:

> "a research process in which practical knowledge is developed whilst in the pursuit of human purposes. It is underpinned by a participatory worldview and attempts to combine theory and practice, action and reflection as well as the participation of other key stakeholders in the identification of practical solutions to pressing issues or challenges in a community" (Reason and Bradbury, 2008).

"Action research is a form of research in which a recursive process is applied, i.e. the research goes through a cyclical process of planning, acting on the plan, reflecting on the outcomes, implementing the change and further re-planning" (Coghlan and Brannick, 2014).

"A participatory, democratic process concerned with developing practical knowing in the pursuit of worthwhile human purposes, grounded in a participatory worldview which we believe is emerging at this historical moment. It seeks to bring together action and reflection, theory and practice, in participation with others, in the pursuit of practical solutions to issues of pressing concern to people, and more generally the flourishing of individual persons and their communities" (Reason, Peter, and Hilary Bradbury, 2001, 2005, p.1).

Action research is an approach to support practitioners to find out different ways in order to provide quality within the industry under study. Koshy (2010) provided a list of action research features which was presented in table 3-1.

1- Action research is a method used for improving practice. It involves action, evaluation and critical reflection, and changes in practice to be implemented.	5- Action research can involve problem solving, if the solution to the problem leads to the improvement of practice.
2- Action research is participative and collaborative; it is undertaken by individuals with a common purpose. 3- It develops reflection based on interpretations made by the participants.	6- In action research findings will emerge as action develops, but these are not conclusive or absolute. 7- Knowledge is created through action and at the point of application.
4- It is situation-based and context specific	

Table 3-1: Action Research

As stated before, this thesis selected and used action research process. Action research is adopted rather than the tradition research because it's capability to deal with the practical concerns regarding the information system developments, also gather data in a clear way, support the future considerations, and helps to identify a successful solution (Parkin, 2009; Reason and Bradbury, 2008; Greenwood and Levin, 2007). Based on this view, this process allows the implementation of required changes within a multimethodology framework that

addresses both hard and soft aspects. So, it is the appropriate tool to be used for the practical nature of this research work.

This research has therefore adopted action research as a general methodology through which to proceed through the evaluation of the proposed framework for the development of information systems in an educational as well as business environment. The reason behind adopting action research is that both the researcher and supervisor are from the academic field and would therefore be part of the research work. The entire process of action research presented in Figure 3-1 which is adopted from Kemmis & MC Taggart (2005) Action Research Spiral.

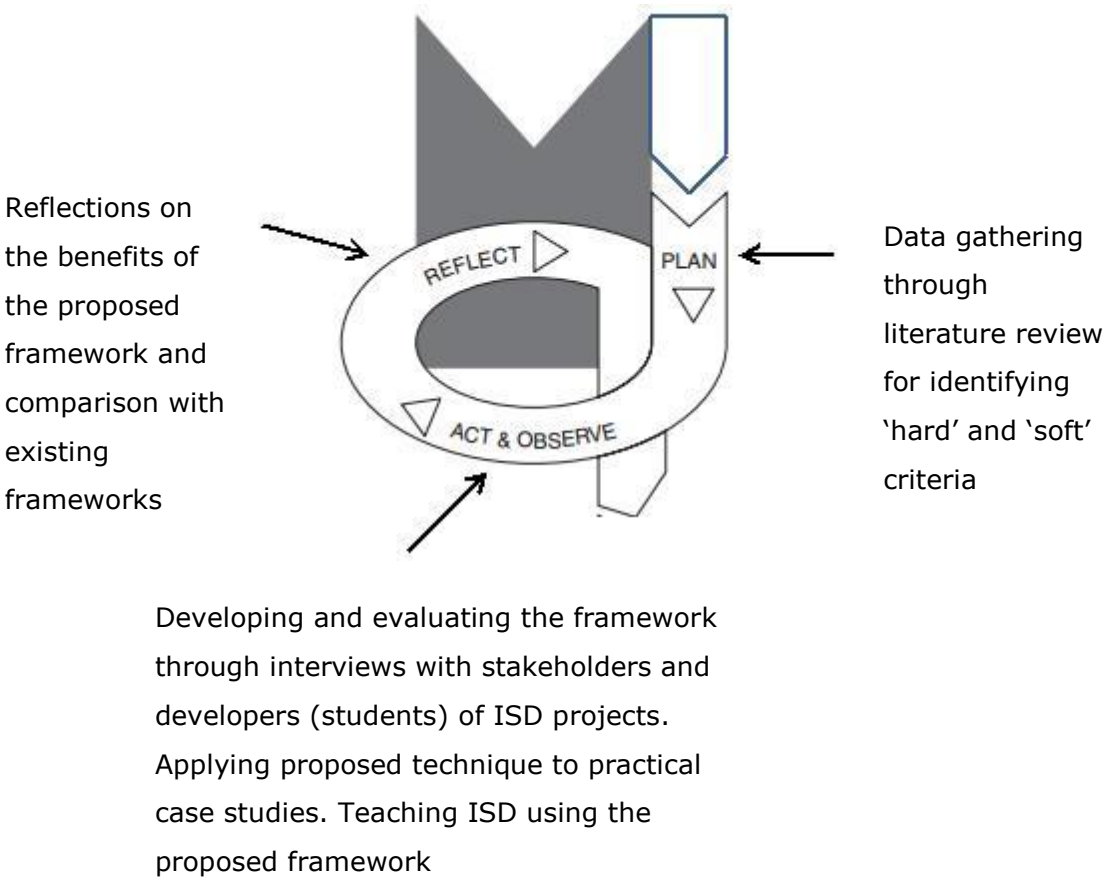


Figure 3-1: Conceptualization of the Research Methodology through Action Research

3.4.1.1 Action Research in the Field of Information System Development

According to Mansell (1991), action research is highly prominent in analysing the issues and performance of information system. This approach to research is efficient at problem-solving activities that adds knowledge and also implements in practice.

Baskerville & Wood-Harper (1996) have observed that action research has been widely deployed in the field of information system development, where researchers have studied IS in organizational settings. Action research is a method that offers potential inputs in improving the practical aspect in the domain of information systems. It has also been utilized in the organizational and educational spectra as a contributing and reliable research method. Action research is therefore, relevant in the context of practice in information system development (Baskerville & Myers, 2004).

3.4.2 Literature Review in Action Research

A literature review of works related to information systems development was undertaken. Issues related to business process, business domain, ISD methodologies, domain-driven design and ISD projects in educational institutes were reviewed. This review shows that there is a need for a multimethodology to handle certain issues related to the system being developed, since these methodologies are categorized separately into hard and soft approaches. The required methodology should be able to handle both hard and soft issues. Ignoring soft issues, and weaknesses in information systems education, are considered to be the main reasons for software system failures. This supports the argument that a multimethodology framework is required and this was proposed in the next step. The literature review presented in Chapter 2 illustrates why the SSDDD framework was proposed.

3.4.2.1 Proposal of Soft Domain-Driven Design Framework as a Multimethodology Framework

The proposed SSDDDF is based on DDD, which is a dominant framework for ISD. A soft layer is added to DDD by combining tools from DDD (UML and implementation pattern) with SSM. The proposed framework is described using the illustrative case study in Chapter 4. The proposed framework is further described and evaluated through different illustrative case studies in Chapter5, and through using it for teaching the module 'Methods and

Modelling' for further reflections on each component of the evaluated framework. The practical case studies and the module 'Methods and Modelling' are described in section 3.4.3.

3.4.2.2 Evaluating the framework by comparing it to others in the literature

This phase in the action research is commenced in the last, proceeded by case studies and interview methods, wherein a comparison between SSDDD and other DDD frameworks is done via literature review research , the feedback gained from case studies evaluation, and feedback from teaching the module 'Methods and Modelling'. The SSDDD framework is compared with other dominant DDD frameworks as it is declared at the beginning of this thesis that DDD is used as the basis for the proposal of the new SSDDD framework as an extension and improvement of DDD. The comparison criteria are formulated based on the consideration of each framework for information systems development. Also, the SSDDD is compared with other four frameworks and methodologies including SSADM, Multiview, SWF, and Agile methodologies to generate a holistic comparison results in order to show the capabilities of SSDDD compared to this sample of methodologies. The comparison and evaluation findings are presented in Chapter 6.

3.4.3 Case Studies Adopted in Action Research

The developed multimethodology framework is applied to real problems by considering practical case studies. The researcher, as a lecturer working in university for many years, and the supervisor also, therefore encouraged the evaluation to take place in the academic environment, in association with different levels of students. Thus, the evaluation of the framework is undertaken as a software development framework. As this environment also enables the evaluation to relate to developers at different skill levels, projects by undergraduate students (junior developers) were chosen to do such an evaluation first, followed by those of postgraduate students (developers). This would allow a good comparison to be made between them and enable clear reflection on the framework for further improvements in the future. For this purpose, two undergraduate projects, 'Peer-Tutoring System' and 'Students' Association' systems are used and presented in the following sections. And another two postgraduate projects 'Peer-Tutoring System' and the 'Schools Liaison Coordination System' are used and presented in the following sections. The comparison between these results will be presented in the last chapter of this thesis. Also, teaching ISD module 'Methods and Modelling' for a group of postgraduate students in

Informatics department is utilized and used as a further evaluation to gain feedback and reflection on each tool of the framework and the framework as a whole.

3.4.3.1 Peer-Tutoring System

In the previous works (Salahat & Wade, 2009) it has been mentioned that a number of information systems were required to support the department, one of which was a peer-tutoring system at the undergraduate level to improve the programming modules. The aim was to design and implement a peer-tutoring system for the introductory programming unit in the Department of Informatics, in order to support the students and reduce the number of failures. One of the problems facing students and lecturers in the university was the difficulty of understanding and mastering the skills required to write and run computer programs successfully. The system was suggested as a means of improving the pass rate at the university and also increasing students' confidence and knowledge when teaching each other during study sessions. Furthermore, this system would reduce workload for lecturers, as the time they spent clarifying a point to a single student could be reduced by enabling students to discuss such points amongst themselves at the tutoring session, thus leaving the lecturer free to concentrate on preparing lessons for the next classes. A number of researchers have suggested that peer tutoring can be particularly useful to support this type of learning because it allows learners to learn and support each other (Goodlad & Hirst, 1989). It is also beneficial in helping students to learn and practice the required skills more actively in a setting that encourages them to be more active and intellectually engaged (Gardner, 1993). Other researchers (Miliszewska & Tan, 2007) have reported the problems of teaching a programming course at Victoria University in Australia and they propose such an approach to enhance the delivery of this module. Xiaohui, H. (2006) raises the difficulties of teaching programming courses in Chinese universities and discusses different modern incorporation strategies to solve this problem; these strategies include concept mapping, peer-learning and e-learning methods. However, the solutions proposed by other researchers show how to overcome the difficulties of teaching programming units by concentrating on delivery methods only, without investigating all the soft and hard system issues involved in solving such a problem (Miliszewska & Tan, 2007; Xiaohui, H. ,2006). In this thesis, a peer-tutoring system is developed using the SSDDD framework to support and improve the teaching process. This solution aims to enhance students' understanding, which may reduce the percentage of failures in this module. The development of PTS as an undergraduate project by junior developers (students studying an IT major) is presented first in section 5.2; this is followed by its further development as a postgraduate project by

MSc Information Management students in section 5.5. These projects are developed using SSDDDF in order to enable an evaluation of the framework as an information systems development (ISD) approach.

3.4.3.2 Schools Liaison Coordination System

The liaison coordination system was another system required as part of the school intranet by the School of Computing and Engineering in the University of Huddersfield. This would be a normal database system to replace an EXCEL one. Students' applications for admission received at the school were being sent to the Recruitment Coordinator on a monthly basis in the form of an MS Excel report consisting of hundreds of records and very precise information for analysis. It was quite time-consuming to analyse this data and to make comparisons against previous years. The school needed the new system to take these Excel reports and generate cumulative reports to provide analysis of applications by grouping them across subject areas, as well as to integrate a contacts database for additional information to compare targeted schools year by year. Section 5.4 will describe how this was undertaken as a postgraduate project by an MSc Information Management student using the SSDDD framework, thus enabling an evaluation of the framework as a software development approach.

3.4.3.3 Student Association System

In Ajman University of Science and Technology, where the current researcher is a lecturer in the IT College, the Student Association System (SAS) is a system required to manage various activities of the Transportation and Student Affairs Departments. The objectives of SAS were to help the students' association to manage and organize students' activities and requirements. These included managing the election process (to choose the association's members) and producing the activities schedule. The SAS system would be managed by the Student Association Department in the university and accessed by many users (university departments and students). Section 5.3 explains how this system has been developed by junior developers (undergraduate students studying an Information Technology major) using the SSDDD framework.

3.4.3.4 Methods and Modelling Module

The module 'Methods and Modelling' is an information systems design module for Masters level students doing MSc Advanced Computer Science and MSc Information Systems Management in the Department of Informatics, University of Huddersfield. This module is taught on 2011 using the proposed framework SSDDD as a further evaluation to gain data

and reflections on the framework it's tools. All of the students arrive on the module with some background in modelling, but those on the MSc Advanced Computer Science course tend to view modelling as high-level programming, whereas those studying for MSc Information Systems Management tend to think in terms of business models. This presents the challenge of moving students into a deeper understanding from different starting points and with different preconceptions about the nature of the subject. For a number of years this module has been taught in block mode over five full days. This mode of delivery was chosen to attract part-time students who were in full-time employment. Over the years, the profile of students on the courses has changed from predominantly working adults to predominantly full-time international students. It became apparent that the intensive nature of block week teaching caused difficulties for this latter group of students, who often arrive in the UK for the first time just a few days before their first class. Restructuring the module to be delivered over a full semester to full-time students presented an opportunity to rethink the modes of delivery and assessment. A 'scaffolded' approach has now been adopted, using an integrated framework, SSDDDF, that has been developed and applied through the few years ago(Steve, W., et al,2012). Section 6.2 present how the proposed framework SSDDD is used for teaching the 'Methods and Modelling' module and how it is evaluated through teaching process as an ISD approach.

Therefore, soft systems approaches were categorized under action research approaches. In this thesis, action research has been adopted through the use of soft system methodology as a guiding methodology for the proposed framework. The use of different cases selected and explored within an educational background and using the framework for teaching ISD has allowed the current researcher, as a lecturer in the educational environment, to act as facilitator and action researcher during the research period. The different techniques used to gather data will be explained in the following sections.

3.4.4 Investigations, Interviews and Discussion in Action Research

3.4.4.1 Using different practical case studies for evaluating SSDDD

Different practical case studies have been used to show how the framework can be used to model and implement business domain processes as a domain-driven design system leading to a software system. Practical case studies have been undertaken by graduate and postgraduate (MSc) students in the school (e.g. School Liaison Coordination System and Students Association System). Following the application of the various stages of the proposed framework, the investigation proceeded by interviewing the different stakeholders

to gain reflections on the benefits of the framework. This included interviewing the developers (students) of the information systems used in the different case studies. For the liaison coordination system case study, the school staff in charge of admission were interviewed. Evaluation and reflections on the application of the framework are presented in Chapter 5. The following sub-methodology was used to accomplish the following:

1- The description of the framework, with its illustrative case study, was explained to students through a workshop in order to guide them in how to use and apply the framework to a real practical case study. This was done for undergraduate groups in one workshop, and for postgraduate students in another two separate workshops conducted at different periods of time.

2- Descriptions of the different case studies were provided to the students and they were asked to start work by applying the framework based on what they had learned in the workshop and from the case study. They could also use different techniques in their investigation and ask advice from the current researcher as facilitator for their projects. The students were given the following case study projects: School Liaison Coordination System, Peer-Tutoring System and Students Association System.

3- The students were asked to reflect on their application of the framework in terms of how it had been used and how it had facilitated their job of developing the information system. They were also asked to record in their reflections any difficulties they had faced during all stages of applying the framework. This was achieved by conducting interviews with the students.

4- For evaluating the proposed framework in School Liaison Coordination System, the school staff (stakeholders) were interviewed, where conclusions were drawn regarding the benefits of proposed system and further improvement options.

The application of the framework, the students' reflections and stakeholders' feedback are presented in Chapter 5.

3.4.4.2 Using the module 'Methods and Modelling' for evaluating SSDDD through Teaching ISD

The module 'methods and Modelling' is for Master students in the Informatics department in the University of Huddersfield. This module has been used to show how the framework can be used to teach ISD and how each tool is used and practised to model and implement

business domain processes as a domain-driven design system leading to a software system. This part of evaluation and reflections is presented in Chapter 6. The following sub-methodology was used to accomplish the following:

1- The description of the module 'Methods and Modelling' and the proposed framework to be used in teaching ere explained to students through a workshop at the begining order to guide them in how to learn and apply the framework to a real practical case study during the period of studying. This was done for all Master students either Msc Information System Management or Msc Advanced Computer Science. Also, methods of teaching and assessments tools were explained in order to let all student to be aware about the assessments tools.

2- A group of practical case studies were distributed to the students and they asked to go through all of them and each student to select one case which he feel happy and comfortable to use it as practical case study during the semester class work. They requested to submit the work at the end of the semester .

3- Frequent in-class surveys were designed and used to evaluate the students' weekly satisfaction. This technique guided the teaching process in order to improve students' learning. This method depended on open-ended questions to obtain the students' feedback. These feedback were considered as a reflections from the students about the tools they studied and practised during the semester.

4- At the end of the course the students were asked to write a short reflective essay including a discussion about the module and how they used the techniques to develop their projects. This technique allowed the students to give their feedback about the techniques that they have been used.

5- The analysing of students final course work to recognise if any mistakes available in their work. The purpose here was to find the reasons behind these mistakes and if they were related to the framework's techniques. This helped the researcher to determine how to suggest an agenda for improving the SSDDD framework.

6- A questionnaire is designed to further evaluate the proposed SSDDD framework as an integrated approach for information systems development. The design of the questionnaire is focused on the various tools of the framework and the contribution of each for achieving more understanding and practical skills.

Teaching the module using the framework, the students' reflections and feedback are presented in Chapter 6.

3.4.5 Combining Evaluation Results from Different Stages of Action Research

Different evaluation methods were used to evaluate the framework as an information system development approach in the educational environment. The framework can also be applicable in the business environment. All the evaluation results are combined and presented in Chapter 7, together with discussions and a consideration of the achievements of this research. The limitations of the research work undertaken and recommendations for future work are also presented.

3.5 Methods for validating the research findings

Gray, Kouhy and Lavers (1995) mentioned that validity as well as reliability are important to be considered by the quantitative study. They determined four important parameters to insure the quality of the qualitative study. These parameters are validity, transferability, credibility and conformability. As the present study is qualitative in nature, the correctness of the results are validated using the following parameters namely validity, reliability, credibility, conformability and transferability.

3.5.1 Reliability

Reliability is used by the research to measure the correctness of the instrument used in data collection (Shenton, 2004). It's important to be sure that the instruments used for data collection is reliable and this can be assured if the instrument can produce a consistent and stable measurement. In this research study, the data is gathered by conducting interviews with students (IS developers) and stakeholders (school staff), and from applying the practical case studies by undergraduate and postgraduate students (IS Developers). Also data is gathered from (postgraduate students) through teaching the module 'Methods and Modelling' using the proposed framework. In-class frequent surveys, Analysis of students class work, Reflective essays, and feedback questionnaire. In this study, it is the responsibility of the researcher to ensure that every respondent will respond the entire questions of the interview or other technique used either at the beginning of the research study or after end of case study application. Also, to be insure that these answers are consistent in all research phases.

3.5.2 Validity

The study must deploy the validity technique to insure that the results obtained by the research study are reflects the requirements of the research (Shenton, 2004). This study managed the validity by framing the interview questions or other techniques in such a way that it contains concepts that are relevant to the research questions and the knowledge explored by the literature review and feedback collected through the application of the practical case studies. The validity concept is important since it will affect the research work finding in a positive way if it maintained properly, otherwise the effect will be a negative one.

3.5.3 Credibility

Donnelly, J., & Trochim, W. (2007) mentioned that the results of the qualitative study can be judged by participants only who can say that they are credible or not. So, the researcher's credibility is the reflector's individual that would judge or predict the credibility of the research. For the qualitative research, the credibility considered the results of such research type as credible or believable. (Patton, 2002). In this research project, the responders are students (IS Developers) and stakeholders (School staff) whom able to judge the results of this research are believable and credible. This will be summarised in their feedback about the application of the framework SSDDD through different practical case studies.

3.5.4 Transferability

Transferability refers to the generalization of the results obtained from the qualitative research. A qualitative research support the researcher by providing them with solid descriptive findings which may be it possible to transfer it to other settings, times and persons and even other kinds of phenomena (Patton, 2002; Trochim and Donnelly,(2007). This research study considered transferability through applying the same framework SSDDDD to develop different practical case studies. The scenario applied is transferred from the first to second, third, and fourth practical case studies.

3.5.5 Conformability

Confirmability is defined by (Trochim and Donnelly,2007) and (Chilisa and Preece ,2005) as the confirmation degree of the results of the study. To achieve this issue, the data collected must be checked and re-checked for many times. For this study, the data

collected from the literature review to initially develop the framework is checked many time to be assure the validity of the soft/hard issues criterial derived are suitable to develop the framework based on them. Also, the data collected after the application of the framework is checked many time to provide proper reflections about the proposed framework SSDDD.

3.6 Ethical considerations

The ethical considerations are very important during the research study since the researcher must protect the participants in the study and the outcome trustworthy is important also to be considered. Silverman, (2013) mentioned that the ethical issues of the qualitative research are very important since the researcher seeking details information through the interviews and other adopted tools, and Creswell, (2013) argued that the ethical considerations must be continue during the whole research project phases. Orb et al., (2001) stated that these considerations are very important because the researcher must assure that he can gained the required access and how to control his effect and behaviour on the participants. The researcher (investigator) must be moral by getting permissions from all respondents and explaining for them the reasons behind this investigation and everything will be treated confidentially either the data or the results of the research Maxcy (2003). According to Ulin et al., (2012) there are three ethical principles must be considered while conducting qualitative research. These are autonomy, beneficence, and justice. Autonomy principle stated that it is up to the participant either to participate or not and we must respect his/her choice, while beneficence refers to the researcher ethical actions to increase the benefits of conducting this research. In the other side, the justice is considered the balance between the benefits issues and the risk of the stakeholders of this research. Finally, it is the responsibility of the researcher to maintain the confidentiality of the results gained from this research and to consider the ethical use of them. Regarding to both universities, they allow these studies since they are part of the curriculum requirements and they concentrated on major point that these projects supposed to go smoothly as other students not included in this study. Also, they requested participants to deal with any used data about the university with top confidential.

This research considered the ethical issues and applied different procedures during the research process. For the undergraduate students in Ajman University, the students who selected me as a supervisor for ” Peer-Tutoring System” and “Students Association System” projects are gathered for a workshop. During the workshop, the projects and the proposed framework SSDDD descriptions were handed and explained to them with other related

issues. The ethical issues were considered and it became clear for all of them that their work is very important and their final grades will not be affected by their opinions about the proposed framework. So, they will apply the framework and their feedbacks are trustworthy and will be considered. Also, the evaluation of their projects and the grades will be given to them by a defence committee not by me only.

The same thing done with the postgraduate students in Huddersfield University and my role was as a co-supervisor for practical projects and teaching assistant while conducting the module 'Methods and Modelling'. The students became confidence to do the work since they became sure that nothing will be negatively affected their work and their final grades.

Chapter 4: Systemic Soft Domain Driven Design Framework (SSDDDF)

4.1 Introduction

The proposed framework (SSDDD) is based on the multimethodology framework, which suggests the combination of diverse methods for the same business intervention (Minger, 2000). It is a multi-method framework that guides the developer through an investigation of a problematic situation and determine its appropriate solution. The purpose of this chapter is to ensure that a comprehensive understanding is achieved for facilitating the modelling and implementation of the domain-driven business processes as an information system. The framework has been developed by appraising and synthesising relevant information from the literature related with different methods and tools used for information system development. It is evaluated through a series of 'action research' case studies, as it incorporates action and reflection through the participation of all the stakeholders. Research cannot be a discrete event, but is a process that has phases with activities to be performed; this research process consists of four generic phases (Minger, 2000).

1- Appreciation: To appreciate the problematic situation and understand the reasons behind the existence of the problem that is faced by actors/stakeholders.

2- Analysis: To analyse the output of the appreciation phase and the techniques used in order to understand how and why they are available.

3- Assessment: To interpret the results and asses different alternatives in order to improve the problematic situation.

4- Action: Recommend changes for improving the current situation by reporting the output results.

For this purpose, the case studies taken are related with the development projects at the researcher's school. The first three case studies focus on the development of a peer-tutoring information system and an information system for the schools 'Schools Liaison Coordination System' and 'Students Association System'. The stakeholders involved are part of the school and participate in daily activities related to the case studies. This chapter will explain the proposed framework in relation to an illustrative case study (peer-tutoring system).

The proposed SSDDD framework (Figure 4-1) is focused on the modelling and implementation of domain-driven business processes as an information system. For developing this framework, SSM is utilized as a guiding and learning methodology with an incorporation of embedded techniques including UML and an implementation pattern (Naked Objects). The development and implementation process is carried out in different stages, which represents the movement from SSM conceptual models to UML use cases. Here, domain-driven design philosophy is adapted to generate 'soft language' as a complement to ubiquitous language that is provided as an input to the stages. The implementation pattern is used after the generation of the final refined change report, which is an input to the implementation process.

The next section presents the proposed framework followed by the evaluation of identified problem using SSM, which consists of three activities equating to the appreciation, analysis and assessment steps of Minger's generic model (Sewchurran & Petkov, 2007). Domain model generation takes place by using UML modelling techniques, since SSM lacks the techniques for taking actions (Sewchurran & Petkov, 2007), and this is equivalent to the action step in Minger's generic model. In this framework, both domain modelling and implementation are equivalent to the action step in Minger's generic model. Thus, the proposed framework satisfies the generic process for conducting action research in business intervention.

4.2 Overview of the proposed framework (SSDDDF)

The proposed framework, as presented in Figures 4-1, 4-2 and 4-3, consists of four phases, where each phase is a composite of several activities. Figure 4-1 illustrates the proposed SSDDD framework, Figure 4-3 represents the conceptualization of the framework, and Figure 4-2 represents the logical processes embedded within it. The peer-tutoring example will be used to show the application of the framework. This case study was suggested and practised by the researcher himself, then reapplied as undergraduate and postgraduate projects under the supervision of the author for evaluating the application of the framework by different developers. There three figures are first demonstrated followed by their explanations.

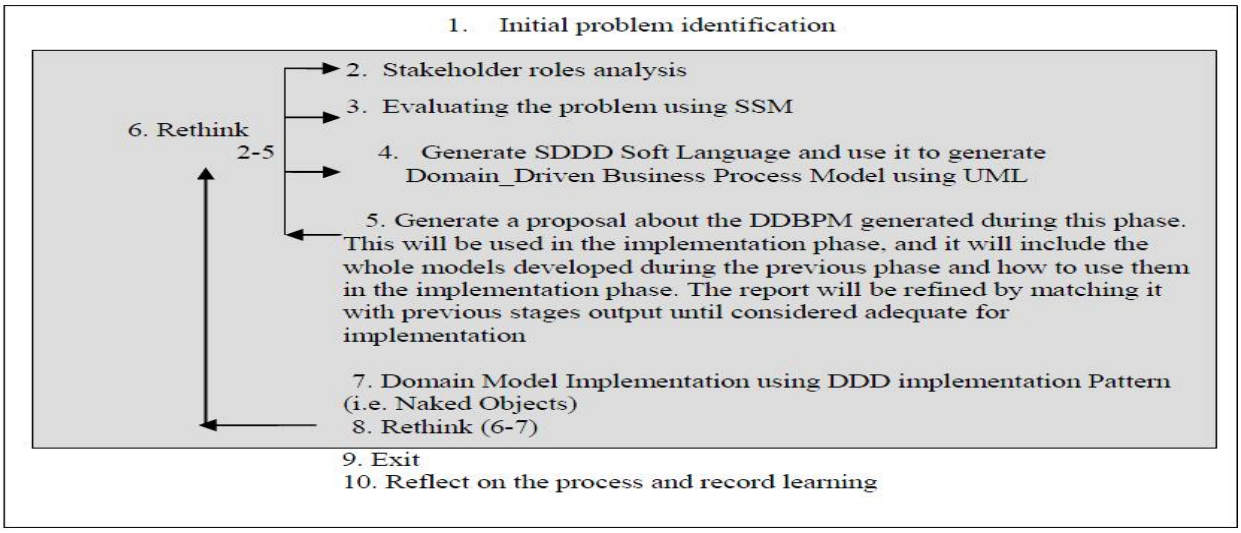


Figure 4-1: The SSDDDF Model

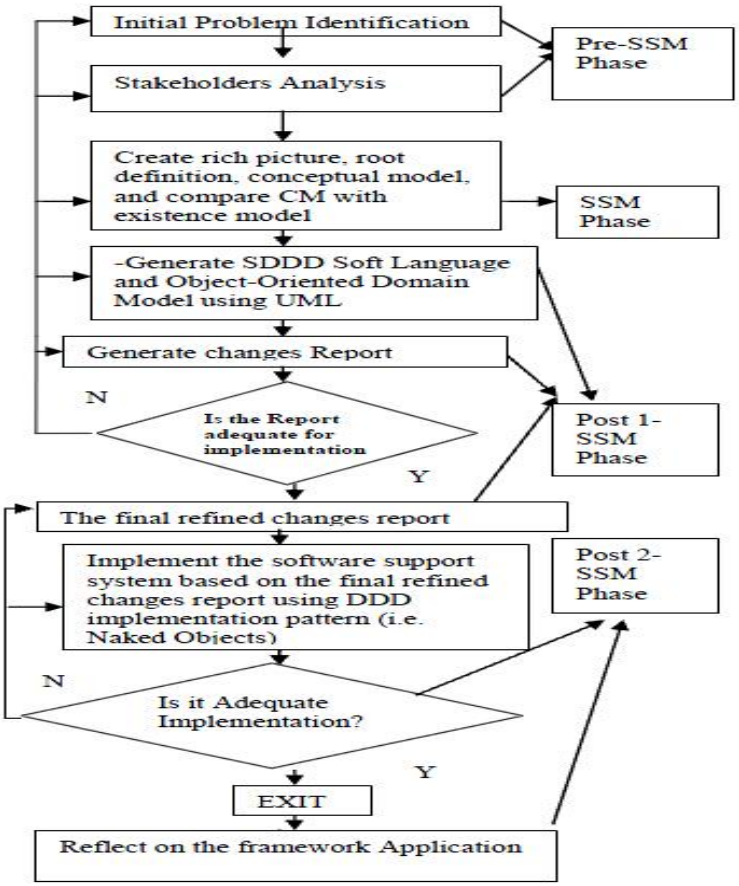


Figure 4-2: SSDDF Logic

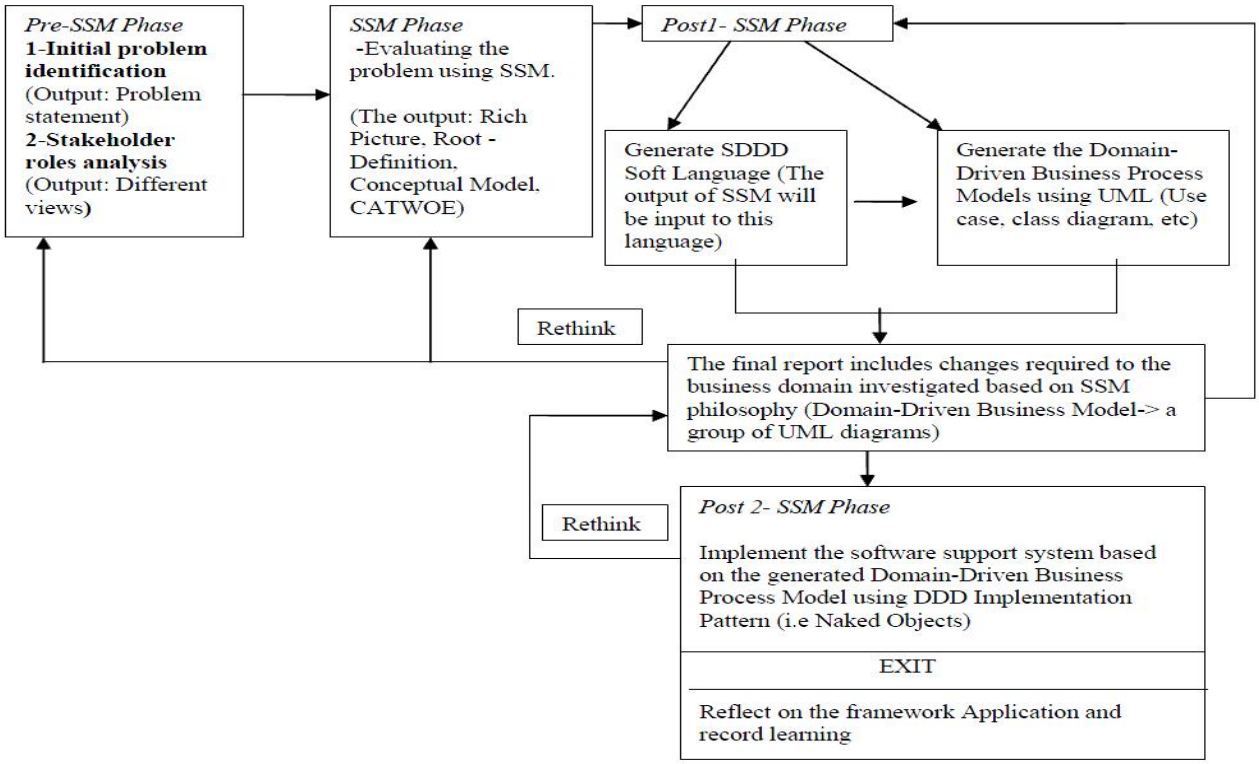


Figure 4-3: The Conception of SSDDF

The details of the above presented framework are explained in the following sections by using the peer-tutoring system (PTS) as a case study as along with exploring other examples from different researches. This case study aims to apply the proposed framework (SSDDDF) to the design and implementation of a peer-tutoring system for the introductory programming unit in the Department of Informatics. The framework integrates soft system methodology (SSM), Unified Modelling Language (UML), and Naked Objects as a domain-driven design implementation pattern. The application of the framework starts with the pre-SSM stage, and then moves on to the SSM application that resolves the problem faced by the stakeholders.

4.2.1 Pre-SSM Phase

Pre-SSM phase includes the identification of the problem and its analysis with the stakeholders. This phase facilitates the SSM investigation of the problematic situation to deploy the implementation of SSM in the school environment. The initial investigation with the determined stakeholders will provide high clarity and understanding to the developers.

In other words, it is expected beneficial to start the SSM investigation based on the results of the pre-SSM stage.

4.2.1.1 Initial problem identification

The problem in a specific area must be determined initially before starting the process of the investigation. Therefore, this stage deals with the initial problem identification, which identifies the roots of the problem and its possible solutions. In the peer-tutoring system case study, the problem is identified to be as follows:

"The problem is focussed on the weaknesses of students in the programming language module, which results in a high percentage of failures. It is proposed that adopting a peer-tutoring system will provide the tutees with extra programming skills that may further reduce the failure percentage".

This initial identification fuels the investigation of next step, which deals with stakeholder roles analysis.

4.2.1.2 Stakeholder roles analysis

The stakeholder role analysis aims to identify the team members of the project along with their roles. Therefore, the roles of all the parties involved in the problem investigation will be clarified to avoid any conflicts and also to facilitate the further proceedings undertaken in the other steps.

For PTS, the following are the needs of the respective stakeholders:

- Peer Tutor – looking for teaching experience, money and reference certificate.
- Peer Tutee – seeking the opportunity for extra help.
- Lecturer – seeking to reduce workload; need to refer weaker students.
- Management – need to reduce failure rate.

4.2.2 SSM Application Phase

SSM is the guiding methodology adopted in the current research. As shown in Figure 4-1, a rethink is involved regarding steps 2-5, which includes the application of SSM for evaluating the problem. SSDDDF techniques are utilized to model the domain's business processes, which is further used to generate a change report including the modelled business domain and its implementation procedure. The output of the SSM stage is offered as an input to the 'soft language' of SSDDDF. Soft language introduced here acts as a complement to the

ubiquitous language of DDD introduced by Eric Evan (2004). It is an 'interpretive ubiquitous language', which includes the output of SSM applications in addition to UL components to facilitate communications between the different stakeholders. This language is therefore, an important part of SSDDDF as it represents the communication tool between the different stakeholders. A detailed discussion on this subject is presented in Chapter 2, in the 'Ubiquitous Language' and 'Alternative to UL' sections. The SSM application phase consists of the following steps:

4.2.2.1- Investigating the problem situation using rich picture model

A rich picture is a drawing that graphically illustrates the issues expressed by people, change processes involved in a resolving those issues, the consequences of changes on the people or stakeholders, the working climate, and conflicts and structures within the change process (Williams, 2005). Anything can be included in a rich picture, where it is used to support the overall understanding of the organisation's situation, goals and structure, and the emerging issues and their repercussions.

Thus, the purpose of drawing a rich picture is to informally capture the main entities, structures and several views of the investigated domain, including stakeholders, operational processes and the connection between these artefacts . A rich picture must be rich with information to assist a person, who may or may not be an outsider, in understanding the complexity of the situation captured during the enquiry process (Checkland & Poulter, 2006, p.24-26). The following figure (Figure 4-4) is a rich picture of PTS drawn initially.

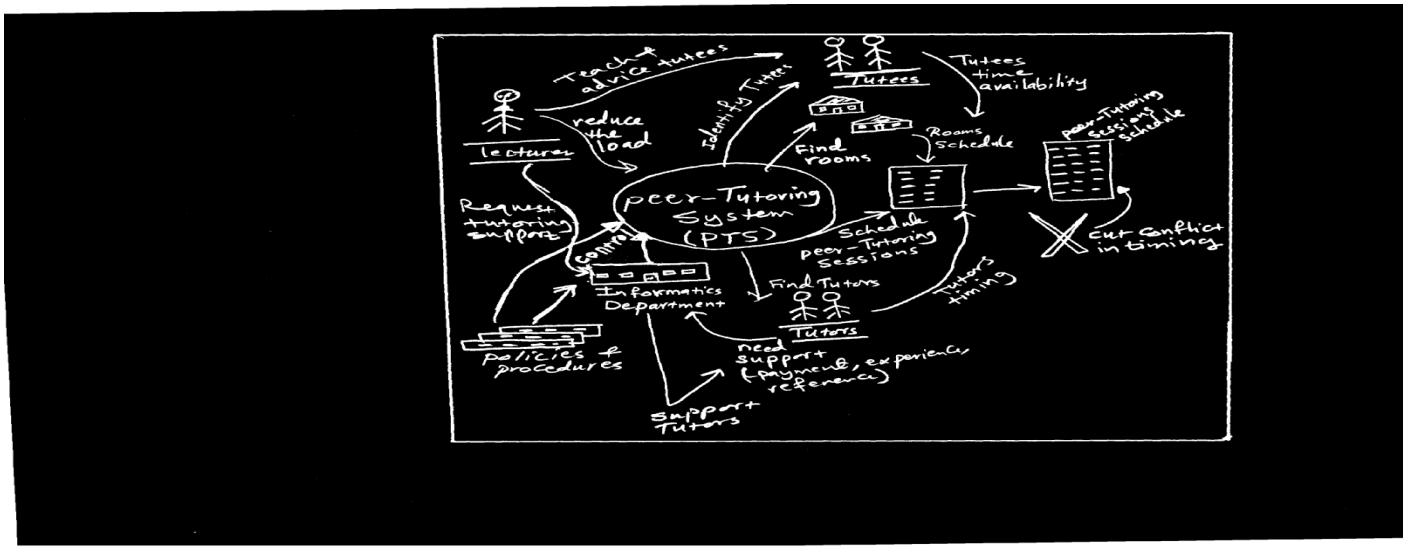


Figure 4-4: PTS Rich Picture

Another example is presented in Figure 4-5, which portrays a rich picture of the student accommodation system in a university (Lewis, 1992).

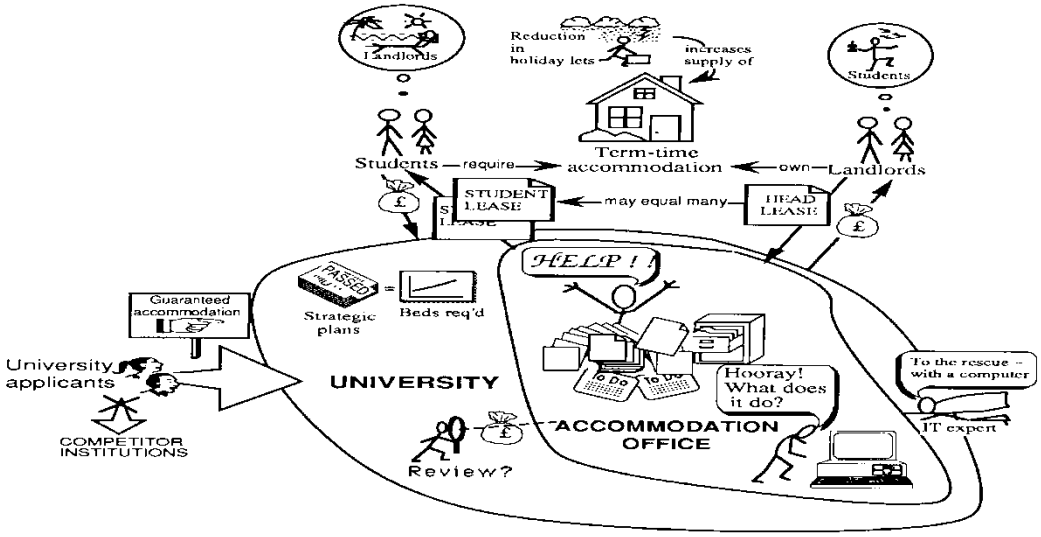


Figure 4-5: Rich Picture of Student Accommodation System

4.2.2.2- Modelling the relevant system using root definition

Root definition (RD) may be described as: "a short textual definition of the aims and means of the system to be modelled" (Rose, 2002). Root definition is used to determine the purpose of the system, which is built from the comprehension of different parties' perspectives regarding the expected functions of the system. In other words, the functionality of root definition is to explore the problematic situation of the business domain based on different stakeholders' views. Modelling a system with the assistance of root definition has been described as a movement from the real world to the perceptions of systems about the real world (Checkland & Scholes, 1990). Williams (2005) mentions that during the root definition stage, viewpoints from different stakeholders are drawn out from the rich picture and presented within a structured development process. According to Jeremy Rose (2002), the format of a root definition is as follows:

> "A system to do **X**, by (means of) **Y**, in order to **Z**"

This format will allow the investigator to understand "*what* the system will do, *how* it is to be done, and *why* it is being done". The following is an example of a root definition taken from a hand out by Jeremy Rose (BIT Department, Manchester Metropolitan University):

> "A university owned and operated system to implement a quality service (**X**), by devising and operating procedures to delight its customers and control its suppliers (**Y**), in order to improve its educational products (**Z**)".

The conceptual model(CM) is derived from RD which will be used to represent the human activity system (HAS) or model. Sometimes HAS derived from the consensus primary task model (CPTM). This model represents natural activities, some of which can be implemented as an information system while the others cannot.

The initial root definition of the peer-tutoring system (PTS) is identified as follows:

"To develop a peer-tutoring system for the Informatics Department for selecting peer-tutees and peer-tutors, scheduling the times of tutoring sessions based on the availability of rooms, tutors and tutees, managing the benefits of tutors and reporting the progress of tutees to the department in order to increase the self-confidence of first year programmers and reduce failure rate within the availability of resources required".

The next step is to test the root definition through Checkland's mnemonic CATWOE (Customers, Actors, Transformers, Worldview, Owners and Environment). The testing for PTS is given below:

C – Customers: People (tutors and tutees) who will be affected by this PTS system.

A – Actors: People involved in this project (current researcher and supervisor).

T – Transformation: Shows the movement from input to output. In this case, the output is the peer tutoring system that is to be used by the students.

W – Weltanschauung (world view): This presents the perceptions taken from the root definition addressing the worth of the current project. This project represents the users' views about the system's benefits and negative feedback.

4.2.2.3- Modelling the relevant system using the conceptual model

A conceptual model is an abstract representation of concepts (entities) and terms, which also determines the relationships between them. The purpose of a conceptual model is to convey the meanings of the concepts and terms used by the domain experts. It further aims at identifying the true relationship between these concepts. The conceptual model, also referred as the consensus primary task model (CPTM), is extracted from the root definition and therefore, represents different stakeholders' views. The model works as a foundation through the conversion from SSM to the UML use cases model. The conceptual models for PTS are presented in Figures 4-6, 4-7, 4-8, 4-9 and 4-10.

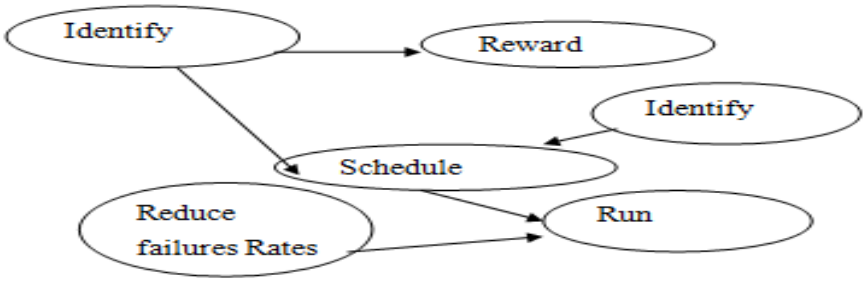


Figure 4-6: CM of Management View

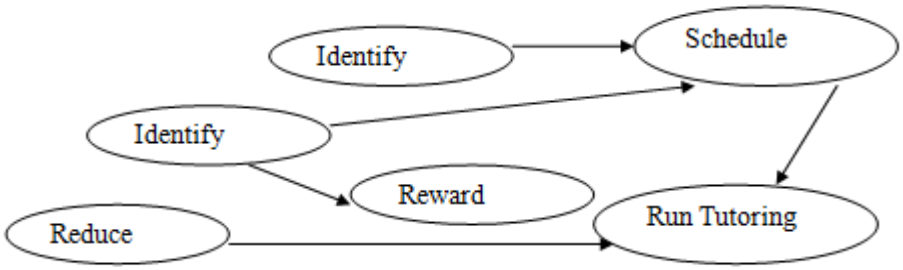


Figure 4-7: CM of Lecturer's View

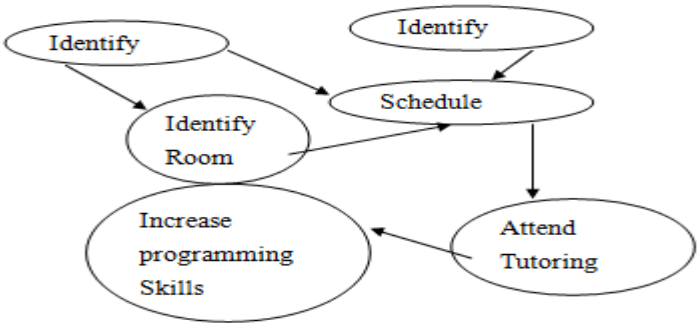


Figure 4-8: Tutees' View

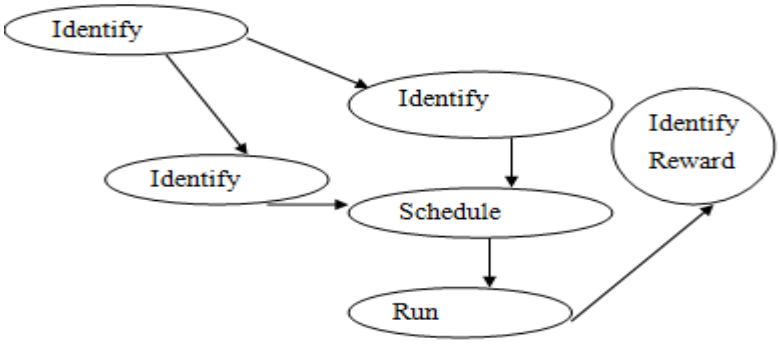


Figure 4-9: Tutors' view

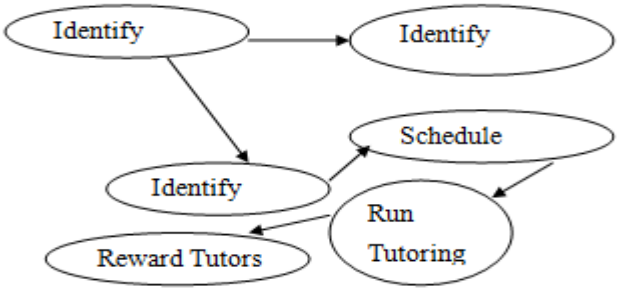


Figure 4-10: Combined CMs (CPTM)

4.2.2.4- Comparing the CM with the real world

The conceptual model, as an abstract representation, has to be tested for validation by forming a comparison with the real world (the current organizational process). The comparison utilizes the activities, organizational goals, objectives and structure using rich picture, root definition and the conceptual model. If the organization has no business domain process model, then the conceptual model can be used as a basis from which domain model can be created (Bustard, Dobbin & Carey, 1996).

In the current scenario, for PTS, there is no real world model to use in comparison with the one being developed. In this case, the developed conceptual model is considered as the real world system model under investigation. This will be used later on as a basis for modelling the PTS system using UML tools.

4.2.2.5- SSDDDF soft language development

Soft language is the first output of SSDDDF. It consists of all the documents and diagrams representing the business domain, and functions as a communication tool between different

stakeholders. The proposed framework revealed that the models developed using the pre-soft systems methodology (Pre-SSM) and SSM phases could provide useful input to the process of developing a soft language (SL). SSM helps the developer to gain a deep understanding of different stakeholders' perspectives, which is an essential component of the soft language as it provides the adequate interpretation of the ubiquitous language. In this case, the PTS soft language consists of the following components: initial identification of the problem, stakeholders of PTS, rich picture, root definition, different conceptual models and CPTM.

4.2.3 Post1-SSM Phase: Object-oriented domain modelling using UML

The conceptual model (CM) or consensus primary task model (CPTM) represents a general view of the domain's functionality. The decomposition of the CM into subsystems will take place by using a subsystem description table (Bustard, Dobbin & Carey, 1996) also, each subsystem activity will be represented in an activity description table. There is a close similarity between conceptual model activities and use cases, which leads to a straightforward conversion process. A new elaborating technique is used to examine any activity that has to be converted to a use case; this is represented in Figure 4-11. This chapter demonstrates this technique and its deployment in the illustrative PTS case study. Also Chapter 5 presents the technique through the evaluation of different case studies. This stage consists of the following steps:

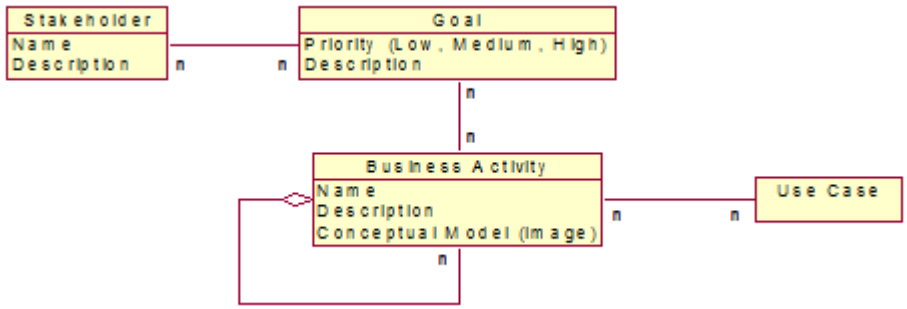


Figure 4-11: Converting SSM Conceptual Diagram to Use Case Diagram

4.2.3.1 Building a subsystem description and activity description tables

A subsystem description table is prepared for each subsystem, which includes a subsystem number, name, heading and activities. Then, an activity description table is prepared for

each activity, including a subsystem number and name, activity name, preceding and following activities, preconditions, input and output data, tasks, business rules and constraints, post conditions, required skills and capabilities, role name and performance criteria. This requirement is essential if the system is large and for simplification needs segregation into subsystems (Al Humaidan, 2006). As PTS is not a large system, there is no need to break the system into subsystems, and therefore the next step is to convert the activities into use cases.

4.2.3.2 Moving from SSM Conceptual Model to use cases

Activities of the conceptual model must be tested to determine their goals; some of the activities can be combined and some can be decomposed. The activities and their goals are tested and mapped to UML use cases as one-to-one relationships. All the use cases are combined in the use case diagram, which consists of use cases and their actors. The use case diagram is a part of the use case model, which represents the organizational business process and forms the basis for modelling the object-oriented domain model. Lastly, all the activities requiring information system are selected as use cases. Based on this process, the following use cases are determined for PTS:

- Add a new peer tutor
- Add a new tutee
- Schedule a peer tutor session
- Calculate money owed to tutor
- Update tutee attendance record
- Identify tutor reward
- Book session

The initial use case diagram is presented in Figure 4-12; this is modified in Chapter 5 on the basis of the new application of PTS during the evaluation of SSDDDF.

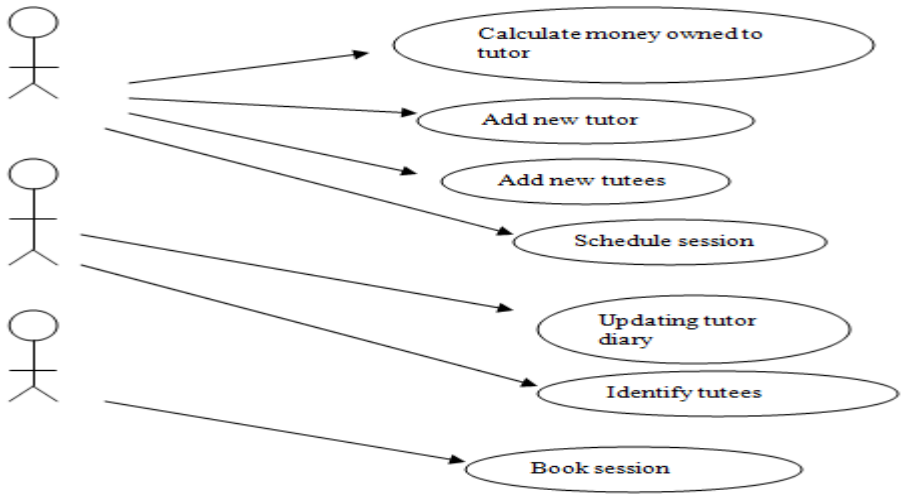


Figure 4-12: Initial Use Case Diagram for PTS

4.2.3.3 Use cases analysis and modelling

The use case diagram presents a hierarchy of business activities by considering the goals of stakeholders. The respective goals highlights the system being requested as per the problem definition during the SSM phase, which needs to be developed. Each use case is described using a textual format template (use case proforma), and is modelled by using UML activity diagram, sequence diagram and class diagram. The activity diagram is used to model the functional, informational, behavioural and organizational system perspectives. The sequence diagram is used to model the interaction between the use case objects (the dynamic aspects of the system). Lastly, the class diagrams represents the static and organizational structures of each use case.

For PTS, the details of each use case are represented by a use case proforma. According to Saraj Din (2009), a use case proforma consist of multiple items' descriptions, which is presented in table 4-1:

1- Number	5-Post-conditions	9-Activity diagrams
2-Name	6-Alternatives and exceptions	10-Supported business processes
3-Primary path	7-Related use cases	11-Activities
4-Pre-conditions	8- Prototype interfaces	12-Notes

Table4- 1: Use case proforma items

However, this format can be simplified if some of these fields are deemed to be unimportant.

The following are the samples of simplified use case proformas for PTS.

Scenario Name: Add a new tutor		ID Number: 1	
Short Description: this use case describes how manager can add a new tutor to the system			
Trigger: management add the information about the new tutor Type: External / Temporal			
Major Inputs:		*Major Outputs:*	
Description	**Source**	**Description**	**Destination**
Tutor name	Administrator	Menu option	System
Tutor ID	Administrator	Details form	System
Tutor Major	Administrator		
Tutor Age	Administrator		
Tutor total Hour	Administrator		

Table 4-2: Add New Tutor Use Case

Scenario Name: Add a new tutee		ID Number: 2	
Short Description: this use case describes how Administrator can add a new tutee to the system			
Trigger: Administrator add the information about the new tutor Type: External / Temporal			
Major Inputs:		*Major Outputs:*	
Description	**Source**	**Description**	**Destination**
Tutee name	Administrator	Menu option	System
Tutee ID	Administrator	Details form	System
Tutee Major	Administrator		
Tutee Age	Administrator		
Tutee total Hour	Administrator		
Tutee Free Time	Administrator		

Table 4-3: Add New Tutee Use Case

Scenario Name : Add new room		ID Number: 3	
Short Description: This use case describes how administrator can add new class room into system.			
Trigger: Administrator adding information about new class room. Type: External / Temporal			
Major Inputs:		*Major Outputs:*	
Description.	**Source**	**Description**	**Destination**
Room number	Administrator	Room number.	Administrator
Room Capacity	Administrator	Room Capacity.	Administrator
Room availability time	Administrator	Room availability time.	Administrator
Room place	Administrator	Room place	Administrator

Table 4-4: Add New Room Use Case

Scenario Name: Create schedule sessions.	ID Number: 4
Short Description: This use case describes how administrator can schedule new tutoring session.	
Trigger: Administrator decide to schedule new tutoring session. **Type:** External / Temporal	

Major Inputs:		*Major Outputs:*	
Description	**Source**	**Description**	**Destination**
Select Subject area	Administrator	List of subject area	Administrator
Select tutor	Administrator	list of tutors able to teach that course	Administrator
Check tutor Times availability	Administrator	Tutor availability Times	Administrator
Select room	Administrator	List of rooms available	Administrator
Session day and time	Administrator	Confirmation Message	Administrator

Table 4-5: Create Schedule Sessions Use Case

Scenario Name: Identify Reward Type	ID Number: 5
Short Description: this use case will identify the reward type for the total time the tutor teach	
Trigger: tutor decide to start teaching **Type:** External / Temporal	

Major Inputs:		*Major Outputs:*	
Description	**Source**	**Description**	**Destination**
Tutor info	Administrator	Updated page	Administrator
Total time worked by tutor	Administrator	give certificate	Tutor

Table 4-6: Identify Reward Type Use Case

Scenario Name: Update attendance record	ID Number: 6
Short Description: this use case will update the student attendance record each time the student is absent or present	
Trigger: tutor find if student is absent or present **Type:** External / Temporal	

Major Inputs:		*Major Outputs:*	
Description	**Source**	**Description**	**Destination**
Update record page request	tutor	Updated page confirm	tutor
Student Name selected	tutor		
Submit to administrator	tutor		

Table 4-7: Update Attendance Record Use Case

4.2.3.4 Developing activity diagrams

Activity diagrams are an integral part of the domain model, which is used to implement the information system. Activity diagrams present the stages of the business process or the software process in a sequential manner. The business process may be carried out by people, software components or computers. Each diagram shows the activities embedded in

any use case within the use case diagram that represents the complete system. The following activity diagrams are the examples from PTS.

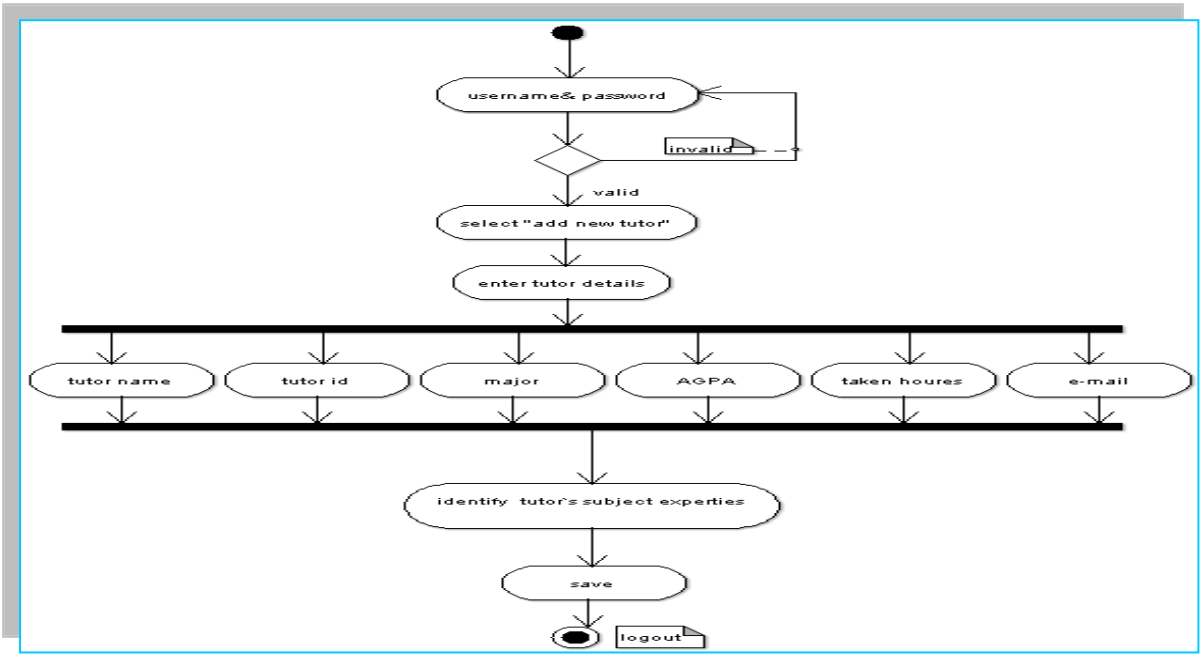


Figure 4- 14: Add a Tutor activity diagram

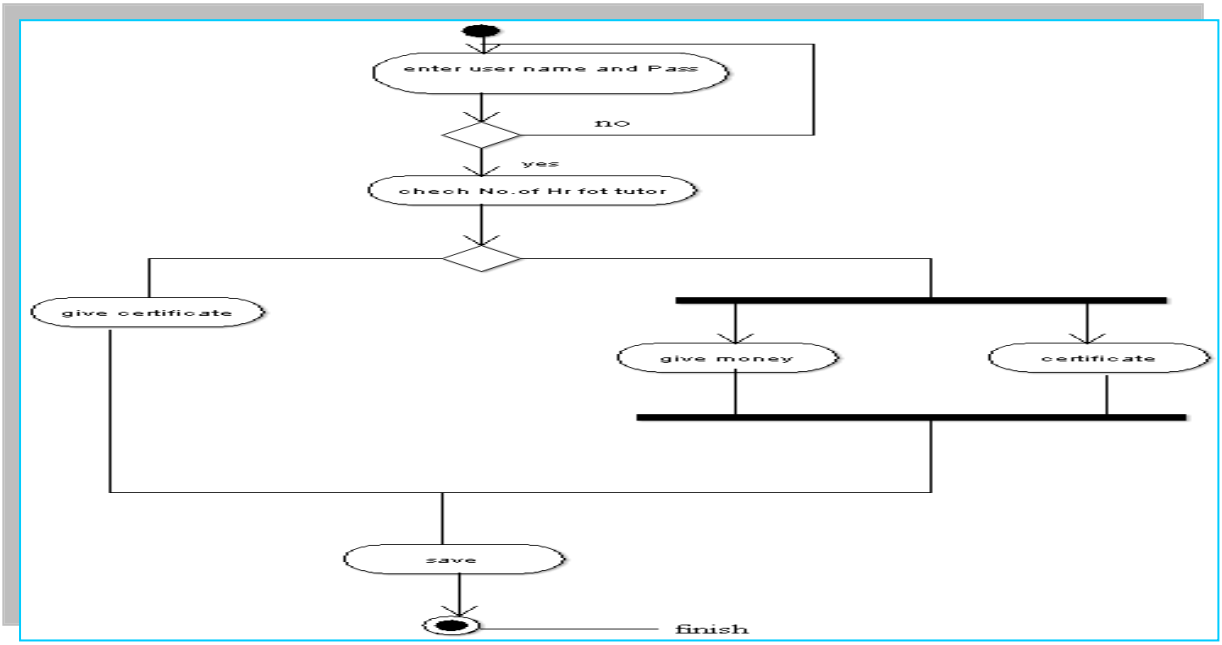


Figure 4-15: Identify Tutor Reward Type Activity Diagram

4.2.3.5 Developing class diagrams

According to Lunn (2003, p.19-20), a class diagram is a collection of all the classes forming a structure of the system. It also demonstrates the relationships between the classes. Class diagrams are developed to model the behaviour of all use cases; these will be combined together in one class diagram called the analysis model, which represents the system in a comprehensive manner (Oliver & Kent, 2009). The resultant model is converted to a design model with the addition of designing aspects required to create the object-oriented domain model. This is achieved by associating the business logic identified in the use cases with classes in the class diagram. SSDDDF considers the class diagram to be a major part of the domain model that can be used to generate the programming code through the implementation pattern. The following is an initial class diagram for PTS, which is modified in Chapter 5 by re-developing PTS as an evaluation of SSDDDF using a postgraduate student project.

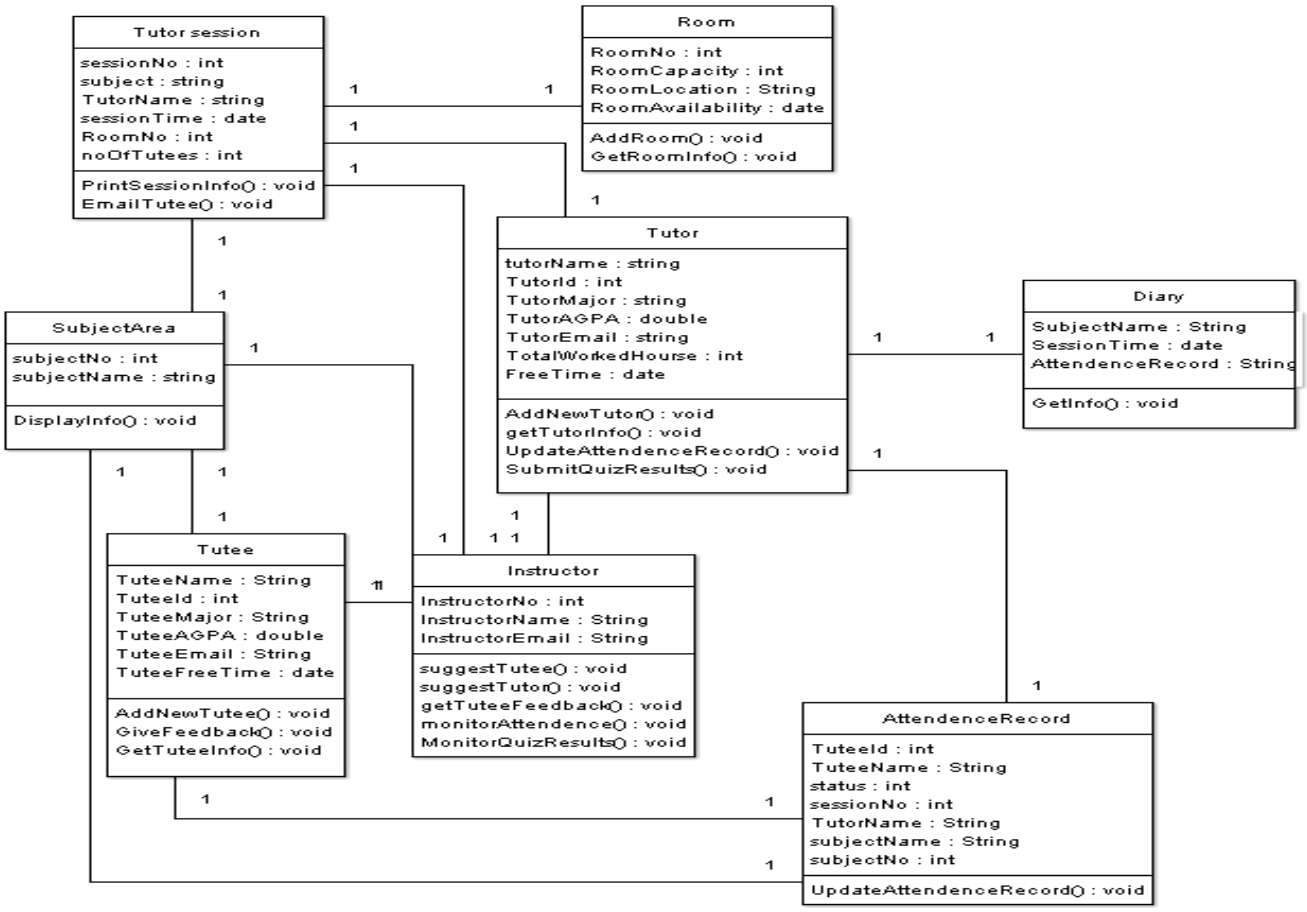


Figure 4-16: Class Diagram of PTS

4.2.3.6 Generating the changes proposal

A 'changes proposal' is generated to improve the domain model, which includes all the models developed during the previous stages as well as guidelines for using them in the implementation stage. The SSDDD framework includes a re-examination of the previous stages to refine the operations performed in Pre-SSM, SSM and Post1-SSM. It is essential to be sure that the exact changes required have already been well-modelled as a domain model. SSM focuses on the generation of the required change report, which can then be recommended for management actions (Checkland & Poulter, 2006; Checkland, 1999; Checkland & Howell, 1998). Thus, the domain model must be modelled, wherein the changes to be made are identified and implemented, and the problems encountered are resolved (Dick, 2002). After achieving this, the PTS, a prototype software, will be ready for further improvements and implementation to serve the programming module. These issues will be discussed in Chapter 5, since PTS is re-investigated by Ucizi Mtenje (2010) as a postgraduate project.

4.2.3.7 Generating the final refined changes report

The report generated in the previous section will be matched against the results of previous stages until an adequate final report is achieved. This includes an evaluation of drawbacks in previous stages that requires modifications and refinements. Finally, the PTS must be monitored and refined to meet the dynamic or new requirements.

4.2.4 Post2-SSM Phase

4.2.4.1 Domain model implementation

The DDD implementation pattern (i.e. Naked Objects) is used in this stage, as it is critical to start the implementation before refining the proposed modelling report. The domain model (mainly class diagram) is used as the prototype for the required information system. However, as per the preferences of the developers, the domain model can be replaced by another adequate implementation pattern such as TrueView. To implement PTS, a Naked Objects implementation pattern is used, though an alternative implementation pattern is presented in Chapter 5.The following are the screen shots of Naked Objects implementation.

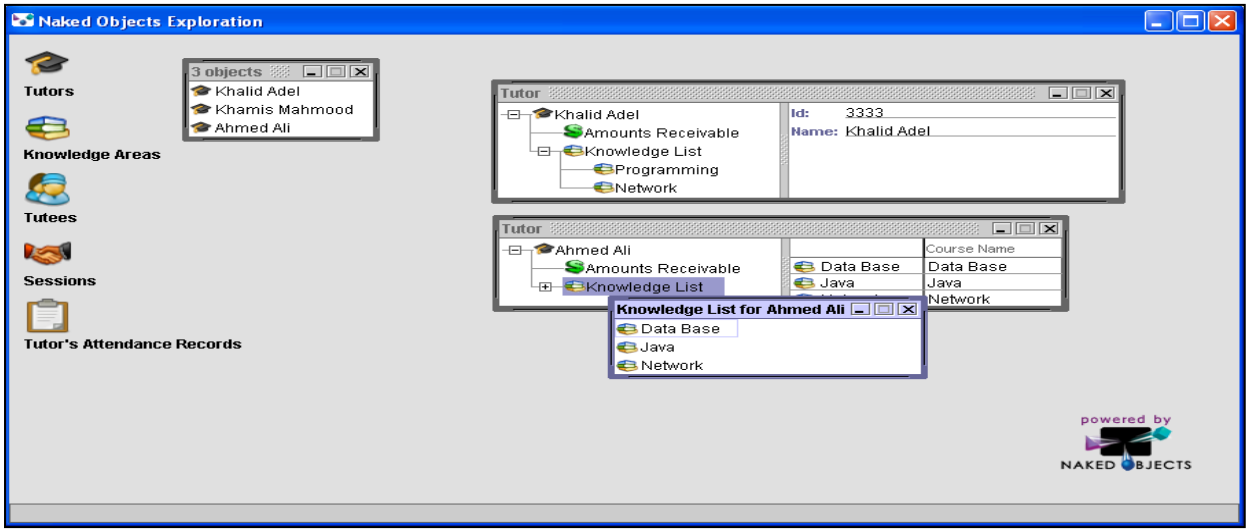


Figure 4-17: Naked Object Implementation - Tutor Attendance

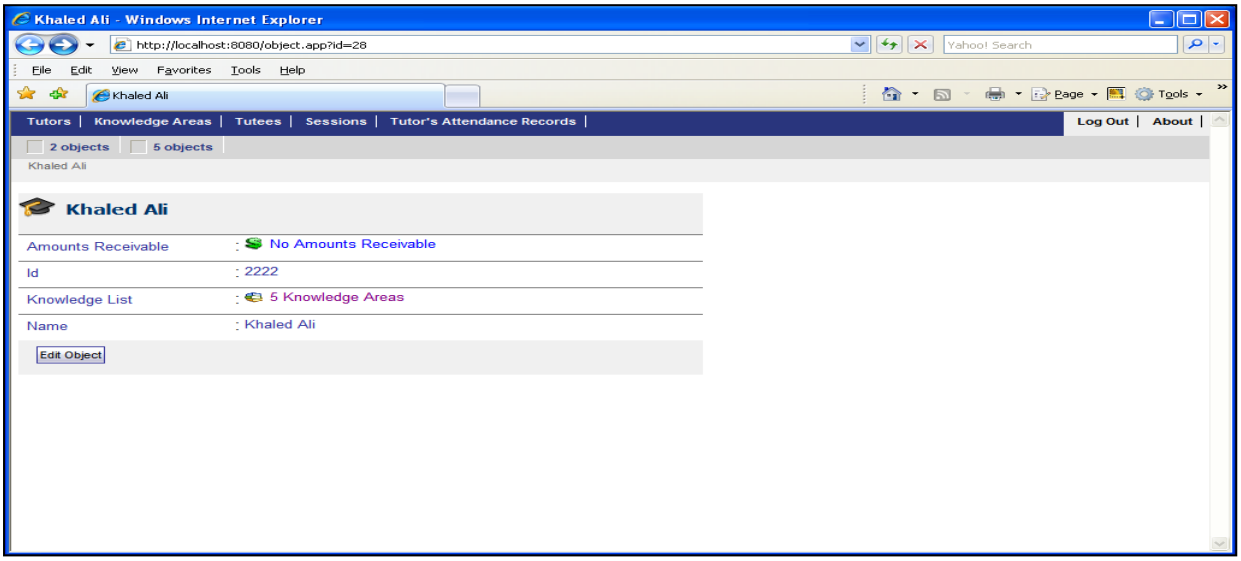


Figure 4-18: Naked Object Implementation - Edit

4.2.4.2 Refining the implemented software system

The implementation results are matched with the refined modelling report and if any deviations arise, changes are made to resolve the emerging issues. This step is presented in the SSDDDF diagram in Figure 4-1 as "Rethink 6-7". For PTS, the implementation must be evaluated by the users (students, tutors, lecturers and administration). Any necessary modification must be incorporated and cross-checked with the requirements based on the logic framework.

44.2.4.3 Exiting and reflecting on the application of the framework

Exit implementation and refinement are executed when an adequate information system has been attained. Then, a reflection on the role of each component of the framework will be done. Reflection refers to the outcome obtained or the conclusions extracted from the actions performed. Finally, lessons learned from combining SSM, UML and the DDD implementation pattern will be recorded to guide further applications. The following section presents reflections and concluding remarks based on the first application of the framework that uses peer-tutoring system as the case study. Further reflections are derived in Chapter 5.

4.3 Concluding Remarks about SDDDF

This work focuses on the proposal and development of a multimethodological framework that can handle both soft and hard issues of business domain process modelling and its implementation as an information system. The new proposed framework is developed based on the idea of domain-driven design (DDD) and soft systems methodology (SSM). A 'soft' perspective has been added to DDD to form 'soft domain-driven design'. The approach can be described as a systemic framework for business domain process modelling and implementation. The framework comprises of guiding steps through various key stages in the development process. It has been evaluated and further developed in an action research program. The example of a peer-tutoring system (PTS) case study has been provided to show how the proposed framework can be applied to a real problem situation. The proposed framework offers the following benefits:

1. It provides a higher level of understanding and clarity to all the stakeholders as the framework successfully applies both the hard and soft requirements. The soft language developed by restructuring and modifying the ubiquitous language facilitates the communication between all the stakeholders and thus provides more clarity. Understanding the business needs and inculcating them in the development of information systems contributes to the successful compilation of the system without any failure. Therefore, the framework performs efficiently as it understands the needs of all the stakeholders and further incorporates changes on the basis of the feedback received at the later stages.

2. The failure of information system emerging due to high complexity, is kept at minimum. As determined by Xia & Lee (2005), the information system is complex as

it addresses both technological challenges and organizational issues, which are not handled efficiently, and thus results in failure. Not only the current framework addresses the stakeholder's views and issues, it addresses the hard components (technological concerns), thus fulfilling all the system requirements. It further follows a systematic approach to fulfill the mentioned objectives, thereby reducing the complexity and information system failures. The previous systems are unable to do so (Xia & Lee, 2015).

3. The framework is effective in managing and handing the changes. As mentioned in the previous sections, the framework comprises of a 'changes proposal' that addresses the dynamic changes and needs of the system and stakeholders as well. The changes are managed in an effective manner with the use of SSM, which used for both general problem solving and management of change. The framework has been most successful in the analysis of situations where there are different views about the definition of the problem (i.e. the views of different stakeholders such as tutors and tutees).

   The existing methodologies were unable to accomplish the same, as discussed in Chapter 2 (literature review). Further evaluations are presented in Chapter 5, and in Chapter 6 the framework is evaluated through comparing it with different ISD frameworks.

Chapter 5: Evaluating SSDDDF as an ISD approach Through Different ISD Projects

5.1 Introduction

While commencing the present research work, the School of Computing and Engineering at the University of Huddersfield was planning to start an information systems development project using SSM and UML techniques within an agile framework to propose recommendations for developing an intranet for the academic school. The department had an operational intranet but this was not widely used, and therefore, professed the need for an inventive method.

For this purpose, an information systems strategy was initiated to investigate to develop the means of developing an intranet that is able to support the university's mission and departmental goals. Initially, use cases were used as the primary fact-gathering technique, but certain limitations in this approach led to a more thorough SSM-based analysis of the situation. It is argued that the techniques of SSM can assist the developers in identifying a richer set of use cases, however the developers with a full use case model still encounter several challenges. The current research emphasises on the object-oriented design and the view that all business behaviour identified in the use case model should be encapsulated as methods in domain objects. Thus, a student object should be a collection of data pertaining to the student details and all the behaviours that may be applicable to a student. Domain driven design refers to these as 'behaviourally-rich' domain objects (Evan, 2004).

A number of software frameworks have been developed, enabling the programmers in constructing prototype applications directly from a behaviourally rich domain model that is implemented in an object-oriented programming language. Prominent amongst these is the Naked Objects framework (Pawson & Mathews, 2002). The Naked Objects framework, as an implementation pattern, has been chosen as one of the SSDDD framework components.

There were different information systems to be developed in the intranet project, two of which were the peer-tutoring system, and a school liaison coordination system for the Recruitment Coordinator in the School of Computing and Engineering (as explained in Chapter3 and 4). The postgraduate students were explained the respective projects, and were allowed to select the suitable one. They were then acquainted with the SSDDD

framework that can be adopted for fulfilling the needs of these projects. The same process was followed with undergraduate students, who had opted to try the framework to develop their graduation projects. All these projects, including both 'undergraduate student projects' and 'postgraduate student projects', were selected for the present evaluation due to the difficulties involved in applying this framework to real business projects amongst the market companies. These projects were explained the methodology chapter 3, in section 3.4.

This chapter presents the gradual application of the proposed SSDDD framework over three years to different student projects at both undergraduate and postgraduate levels. Section 5.2 presents the early stages of applying the framework in the peer-tutoring project, at undergraduate level. Section 5.3 will present the 'Students' Association System', which is an undergraduate project, and section 5.4 presents the application of the model to the postgraduate 'Schools Liaison Coordination System' project for the Recruitment Coordinator at the School of Computing and Engineering. Section 5.5 presents the application of the model to the postgraduate 'Peer-Tutoring System' project, which has also been used as an example while explaining the framework as well as an undergraduate project. The framework is redeveloped here to benefit from the learning process of SSM and to solve the problems of the undergraduate students, as their skills are less proficient than those of a postgraduate developer. These projects are already explained in the methodology chapter3.

5.2 Undergraduate Project: Peer-Tutoring System

As aforementioned in Chapter 3 and 4, action research is used in order to evaluate the framework as a development approach in an iterative manner by using students' projects at different levels. This section describes an undergraduate student project focusing on the peer-tutoring system, in which junior developers/undergraduates have adopted SSDDDF as a development approach. The undergraduates have limited practical experience comprising of their study in university or what they have learned and practised by themselves. This is a group work project and their feedback is used to make further improvements when applying and practising the framework in other projects. Later, the framework is applied as a development approach within a postgraduate student project, using the same and different domain objectives. The other undergraduate project (SAS) is undertaken and discussed in parallel with this project.

5.2.1 Pre-SSM Phase

5.2.1.1 Initial problem identification

The undergraduate 'Peer-Tutoring System' was selected as a group work project by adopting the SSDDD framework. Simultaneously, another project is selected, which is explained in the next section. The methodology adopted in these evaluations is an iteration process that intends to identify the problems encountered in this project and determine solution to support the later projects. Thus, these two parallel undergraduate projects are expected to support the following projects undertaken by postgraduate students. They will learn from the mistakes made by previous students and try to avoid them; this is because at the heart of SSM is an enforced learning process, which is the main purpose of using it as a guiding methodology. Since the current researcher is a lecturer in the IT College of Ajman University located in UAE, he was assigned to be the supervisor of this project, which took place during the second academic semester of the academic year 2008-2009, between 1st February and 1st June, 2009. The group of undergraduate students were asked to use the newly developed SSDDD framework to execute their projects. At that time, the framework was new and had been first published in the Innovation08 conference, in November 2008, at Al-Ain University, UAE. The first version adopted the workflow approach instead of domain driven design, but it was subsequently modified and was presented in the UKAIS2009 conference in March 2009, at Oxford University. The second updated version of the framework was developed at the end of the semester, after considering the feedback of the students, and then submitted to the WASET Conference in Amsterdam during September 2009. The students started the project using the first updated version of the SSDDD framework. Their work and feedback are presented in the following sections.

5.2.1.2 Stakeholder roles analysis

The initial analysis of stakeholders determined the following stakeholders and their roles:

- Peer Tutor – looking for teaching experience, money, experience and reference.
- Peer Tutee – looking for extra help in programming language.
- Lecturer – seeking to reduce workload; need to support students with weaknesses and improve their skills.
- Management – need to reduce failure rate and to support both tutors and tutees.

5.2.2 SSM Phase

5.2.2.1 Investigating the problem situation using rich picture

Any element, representing the actors in a system, can be included while forming a rich picture as there are no specified rules. Different shapes can be used, such as pictures, to represent a particular situation. For example, the crossed swords are used to represent a conflict situation and arrows to show relationships. Based on this, the undergraduate students investigated the problem situation of the peer-tutoring system and came up with the rich picture presented in Figure 5-1 below:

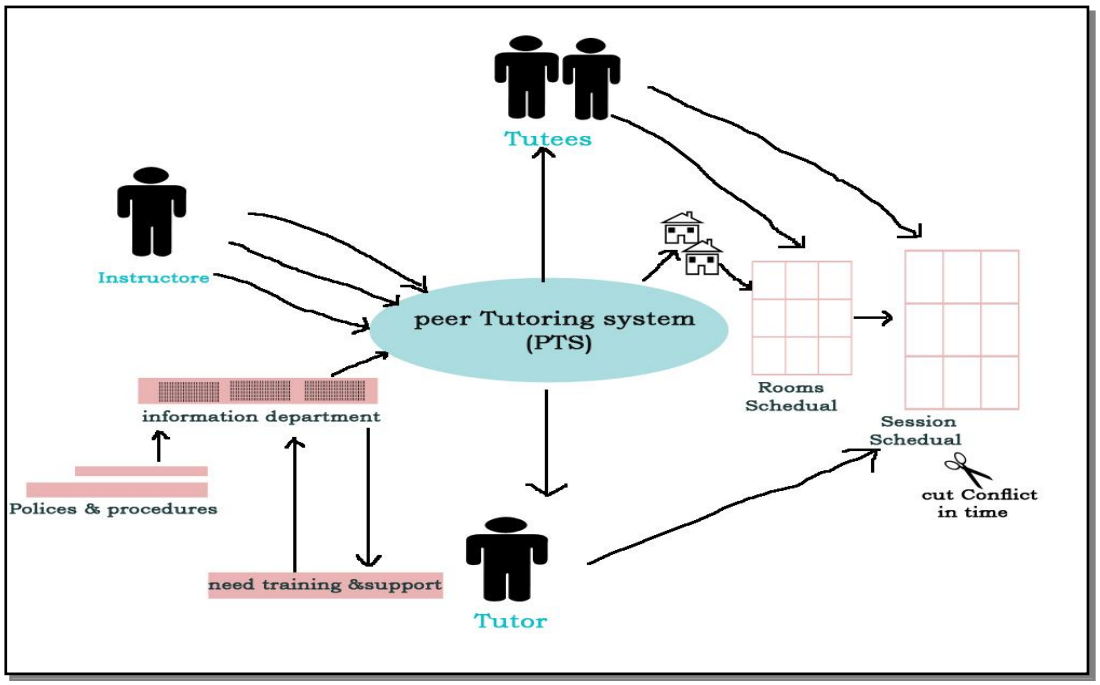


Figure5- 1: Rich Picture of PTS

5.2.2.2 Modelling the system using root definition

Root definition represents the mission of the targeted system and addresses the problem situation from different viewpoints. This is then tested using Checkland's mnemonic CATWOE for specifying the stake holders of the system and their purpose. It is compulsory to identify the root definition according to SSM to explain important issues in the system for commencing appropriate modelling. The root definition is used to construct a conceptual model (CM) or consensus primary task model (CPTM). For the peer-tutoring system, it was identified by the undergraduate students as follows:

"To develop a peer-tutoring system for the Faculty of Information Technology to select the peer-tutees and peer-tutors, to schedule the time of tutoring sessions based on the availability of resources required such as rooms, tutors and tutees, to manage the benefits of tutors and to reduce failure rate".

5.2.2.3 Modelling the system using the conceptual model

This stage is explained in Chapter 4, showing how the root definition is used to extract the conceptual model, which represents the views of different stakeholders. In this case, if the modelled root definition is an accurate representation of the system, then the conceptual model will describe the system activities that might take place. The following conceptual models (CMs) were developed by the PTS group based on what had been done in previous works.

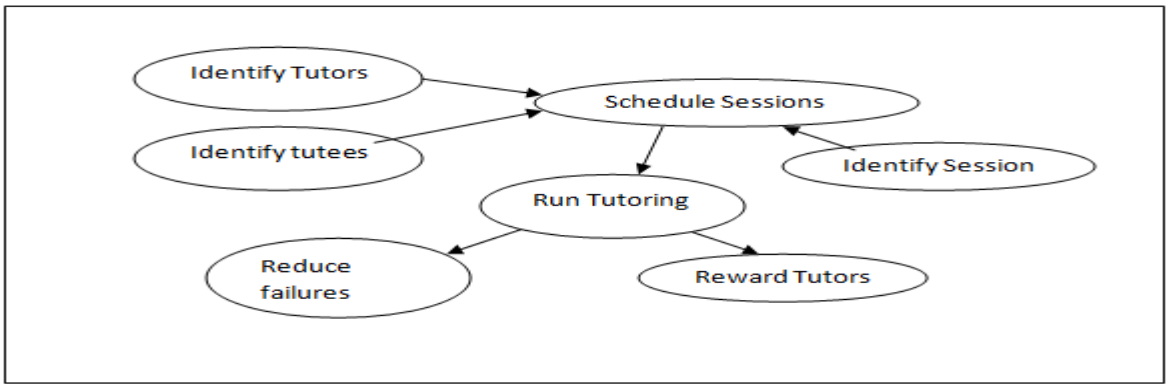


Figure5- 2: CM of Management View

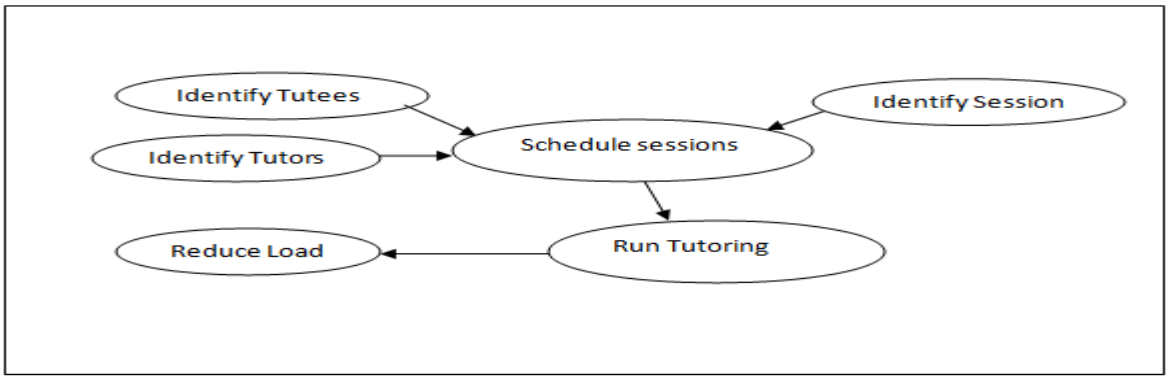


Figure5- 3: CM of Lecturer's View

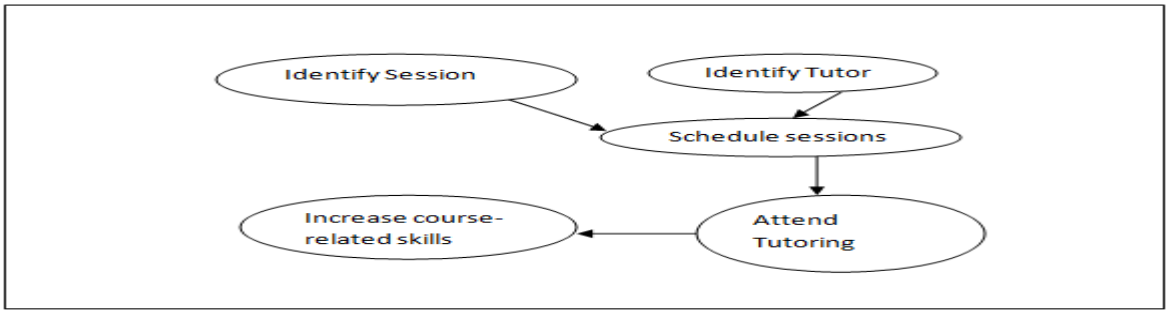


Figure5- 4: CM of Tutees' View

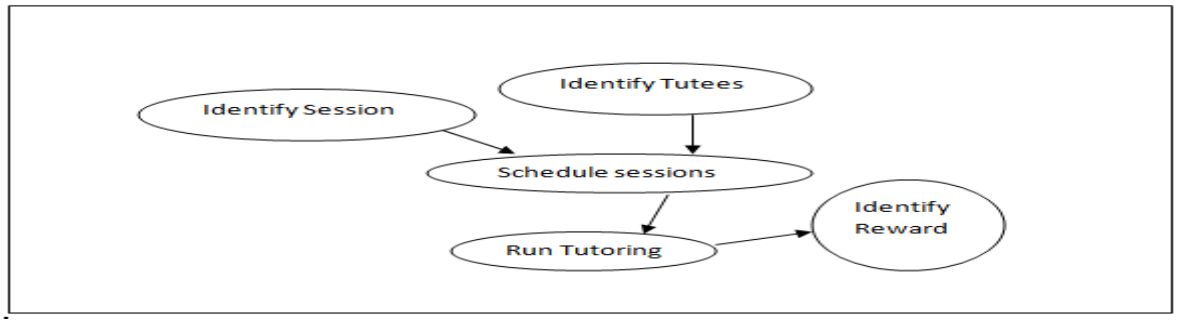


Figure 5-5: CM of Tutors' View

The above models are then combined into one model called the consensus primary task model (CPTM). The CPTM represents the points that are agreed by all. For example, corresponding to the need to schedule peer tutor sessions, stakeholders might have different priorities about optimum times but all reached to the same conclusion. Other examples of consensus points include the need to identify peer tutors (volunteers, best students, future teachers), the need to identify tutees (volunteers, or refer weakest students) and the need to reward tutors (money, good references, separate certificate, credits).

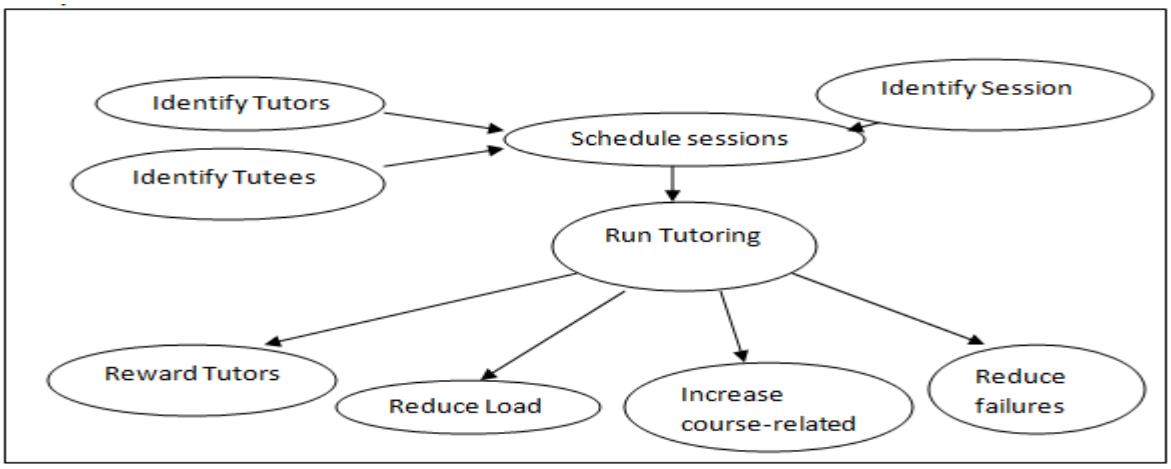


Figure5- 6: Consensus Primary Task Model (CPTM) for PTS

5.2.2.4 Comparing conceptual models to the real world

The developed conceptual models were considered as actual system models because PTS was not yet available and so there were no real life models available to compare with the above developed CMs. The SSDDD framework describes the role of soft system methodology, which requires the investigator to compare conceptual models with actual real world models. As in the present case, there is real world system, the developed conceptual model is used as the real world system model. Therefore, the students used the developed conceptual models as a basis to model the PTS as a domain model. The other output models from SSM and the CPTM are the major components of soft language and were used to generate the domain model.

5.2.3 Post1-SSM Phase: Moving from Soft Language (SSM Phase) to Domain Model

5.2.3.1 Moving from SSM conceptual model to UML use cases

The SSDDD framework has adopted UML to model the domain model. For this purpose, the conceptual model is first converted into use cases and use case modelling. The extracted use cases are then used to develop a UML sequence diagram, class diagram and activity diagram. The next subsection will show the conversion from CM to use cases.

1- Use case derivation from conceptual model

The combined CMs presented in the CPTM were converted to use cases using the conversion method explained in Chapter 4. All the activities requiring information system were selected as use cases. The following use cases were identified:

- Create/ adjust a new peer tutor
- Create/ adjust a new peer tutee
- Schedule a peer tutor session
- Insert a tutor attendance record per session
- Calculate amount receivable by tutor

The use case diagram which the students created for PTS is presented in Figure 5-7; the preparation of this was based on SSDDDF, as explained in Chapter 4.

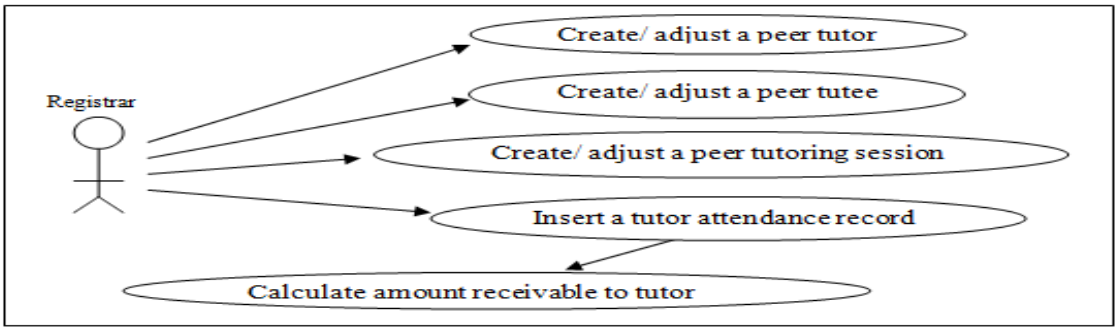


Figure5- 7: Use Case Diagram for PTS

2- Use cases analysis and modelling

The undergraduate group work projects relied on the concepts determined in the framework, according to which the use case diagram presents a hierarchy of business activities by considering the goals of stakeholders. This further highlights the system being requested and must be developed according to the problem definition during the SSM phase. In addition to the textual format template (use case proforma), the use case is modelled using a UML activity diagram, sequence diagram and class diagram. Whereas the purpose of the activity diagram is to model the system perspectives, the sequence diagram is used to model the interaction between the use case objects (the dynamic aspects of the system). Also, the class diagram is prepared to present the structure for each use case,

which is at the end of the system structure. For PTS, each use case is presented by a textual format template, called a use case proforma, which shows the details relating to it. Chapter 4 describes the structure of use case proforma that is presented in table 4-1 prepared by (Din, 2009). The Appendix 2 represents the samples of simplified use case proformas for PTS.

3- Generating activity diagrams based on use case diagram

The student group created the following sample activity diagrams, based on the use case diagram, to represent PTS.

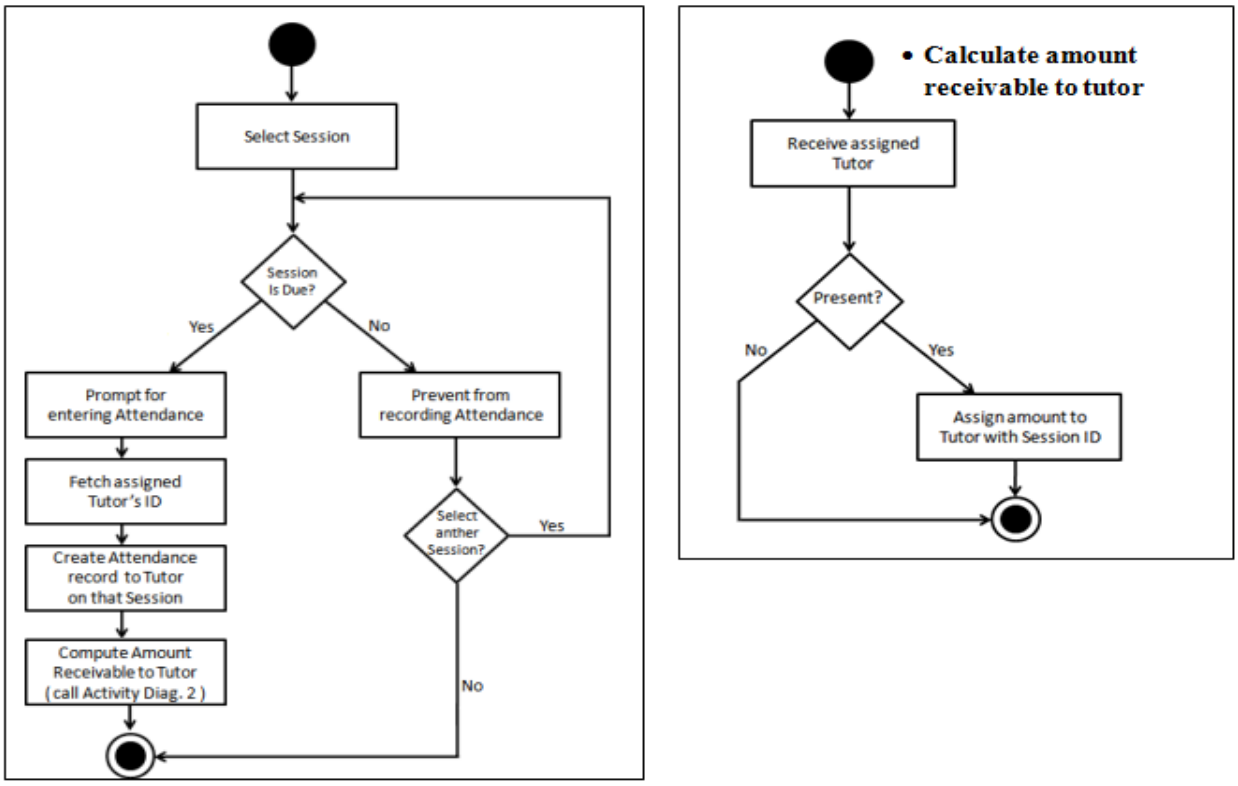


Figure5- 8: Activity Diagrams

5.2.3.2 Generating the class diagram based on use case and activity diagrams

A class diagram is "a collection of all classes and the relationship between them, and defines the static structure of the system" (Lunn, 2003, p.19-20). The students in the undergraduate group reported that the domain classes were understood, and the following class-level specifications with their associations were derived.

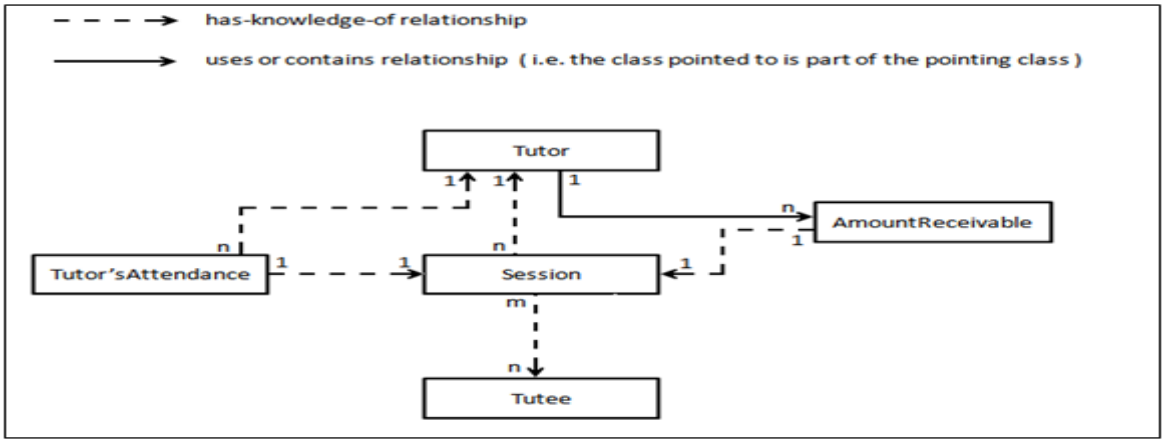


Figure 5-9: Class Association

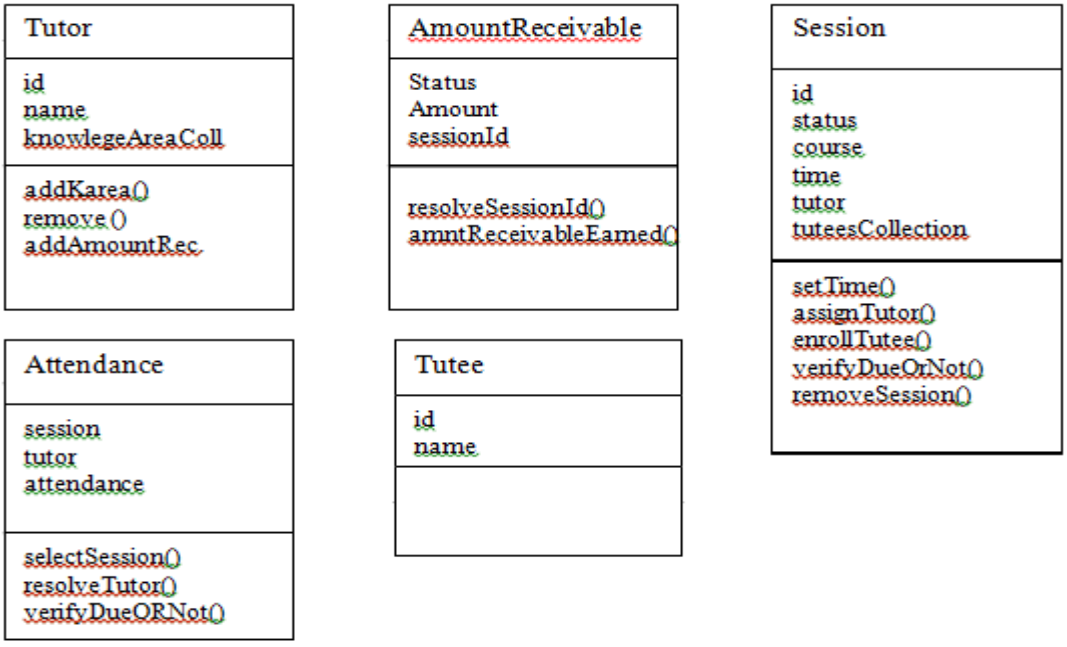


Figure5- 10: Class Level Specification

5.2.4 Post2-SSM Phase: Software Implementation

The domain model is the base from the programming code using the Naked Objects implementation pattern is extracted, which is recommended by the SSDDD framework. Naked Objects is adopted here since it supports the creation of system user interface from the business domain model. After a brief description of the implementation of PTS using the Naked Objects implementation pattern, the students applied it, which is presented in Appendix 3. An evaluation of the implementation, and a reflection on the framework as a development approach, are provided in the next section.

5.2.4.1 Implemented software evaluation

The students reported that peer tutoring is a widely implemented concept deployed through different methods across different universities in the world. PTS is a promising application if it can be adopted effectively in the university. Insight has been gained into the open source packages and fully-committed community (Java, Eclipse, Naked Objects, etc.), which can open wide horizons for future work, and hence careers. A close experimental understanding of the underlying software structure has also been achieved, as along with an awareness of the requirements of the software framework and the related benefits.

5.2.4.2 Reflection on the SSDDD framework

The benefits gained from the adoption of SSDDDF framework have been mentioned by the students as:

- Clearer requirements definition through investigation using the soft system methodology (SSM);
- High commitment to the object-oriented approach using UML and the Naked Objects framework;
- Shorter project lifecycle as requirements are clearly identified from the beginning, thanks to SSM.

This reflection, based on the students' achievements, supports the arguments for using the proposed framework as an information system development approach to understand soft and hard issues of the system being investigated. The students stated that the system requirements were clearer for them because of using SSM at the beginning, which reduces the time required for development of information system. This evaluation and others will be further discussed.

5.3 Undergraduate Project: Students' Association System

The above section (5.2) describes an undergraduate project on the peer-tutoring system, which was done using the SSDDD framework and undertaken as a group work project in parallel with this one..

A group of undergraduate students in the IT College of Ajman University, UAE, selected the development of a Students' Association system (SAS) as their graduation project topic during the second semester of the academic year 2008-2009, between 1st February and 1st June, 2009. As mentioned before, the current researcher was assigned as the supervisor for that project, and asked the group to use the newly developed SSDDD framework to do

it. This framework has been discussed in detail in Chapter 4 and briefly in the previous section, 5.2. The students started the project using the first updated version of the SSDDD framework. Their work and feedback are presented in the following sections.

5.3.1 Pre-SSM Phase

5.3.1.1 Initial problem identification

The students reported in their project that the Scientific Student Association in Ajman University of Science and Technology required a system to solve the problems that they were facing in their work. From the different stakeholders' views, they identified the key problem areas that need adequate attention. They were the need to simplify the election process for the association's members, to offer easy communication between student members, and to produce the activities schedule and also organize them. The next section will show the different views of stakeholders as reported by the students in their project.

5.3.1.2 Stakeholders roles analysis

The following stakeholders with their corresponding roles were identified:

- Association management – looking to organise student activities, parties and journeys.
- Association member – a chosen student who is responsible for communication between students and management, and who receives complaints and suggestions.
- Students – looking for opportunities for activities and management attention to their needs.
- Student Affairs – needing to manage and supervise the activities process in the real field, and keep in touch with students doing training outside the university.
- Colleges – seeking to maintain contact with the association's management to submit their requests and schedules (such as seminars, scientific trips or graduation parties).
- Transportation – needing to make transportation facilities available based on the activities schedule provided.

Table5- 1: SAS Stakeholders and their roles

5.3.2 SSM Phase

5.3.2.1 Investigating the problem situation using rich picture

The concept of rich picture and its definition has been explained in the previous sections. In SAS, the commonly used elements are the actors of the system that are presented by different shapes.

Accordingly, the undergraduate student group investigated the problem situation of the Students' Association system and came up with the rich picture presented in the following figure (5-12).

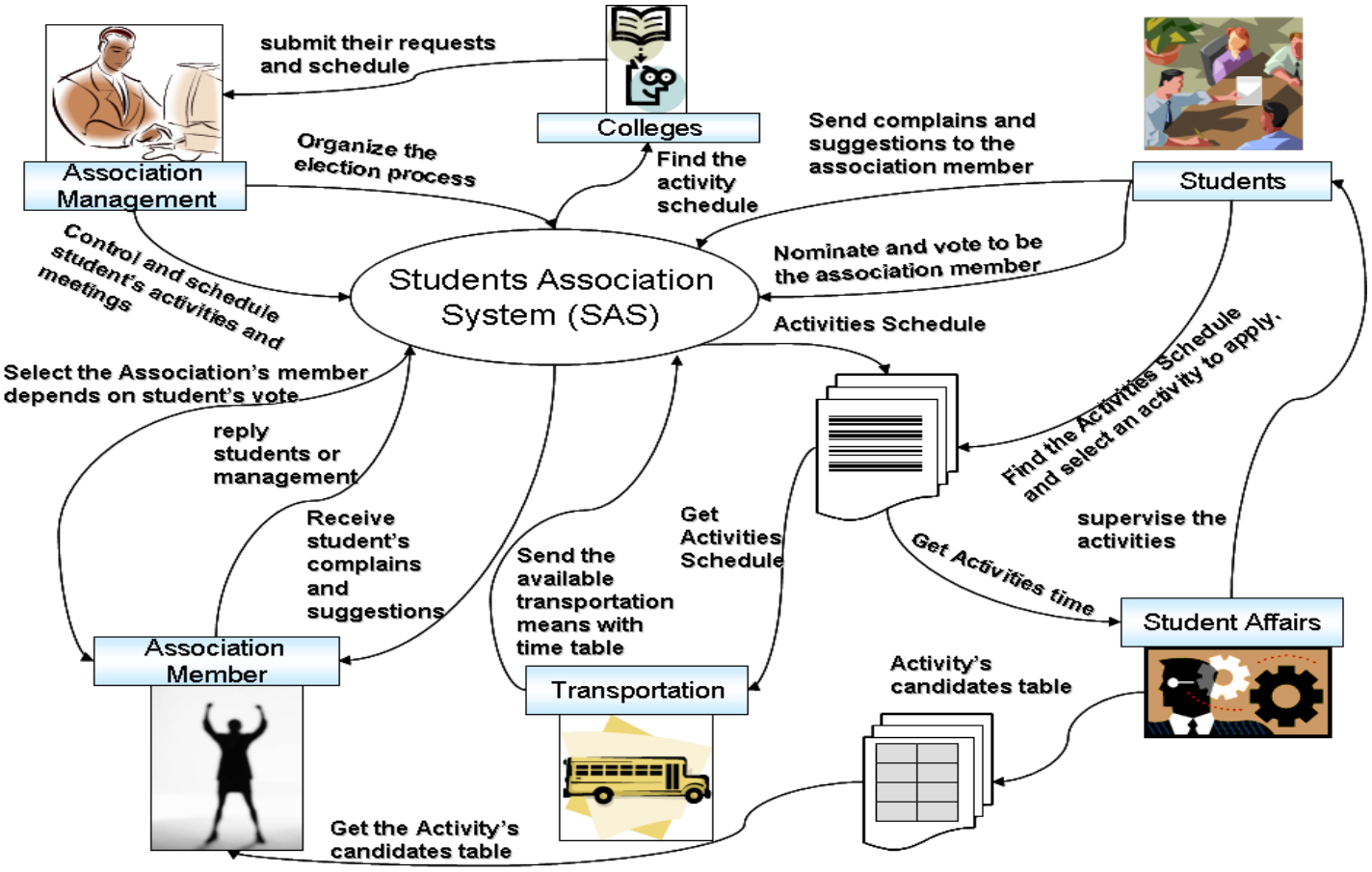


Figure5- 11: Rich Picture of SAS

5.3.2.2 Modelling the system using root definition

The explanation of root definition has been performed in the previous case study. For SAS, the root definition was identified by the student group as follows:

"To develop a Students' Association System for the Students' Association Department to control and schedule students' activities and meetings, organize the election process, select the association members depending on students' votes, set the activities schedule and manage communication between students and management through the association members".

5.3.2.3 Modelling the system using the conceptual model

Root definition is used to extract the conceptual model, which represents the different views of stakeholders. In this case, if the modelled root definition is an accurate representation of the system, then the conceptual model derived will describe the system activities that might take place. The following conceptual models (CMs) of SAS were developed by this student group based on in the activities of the previous works mentioned above.

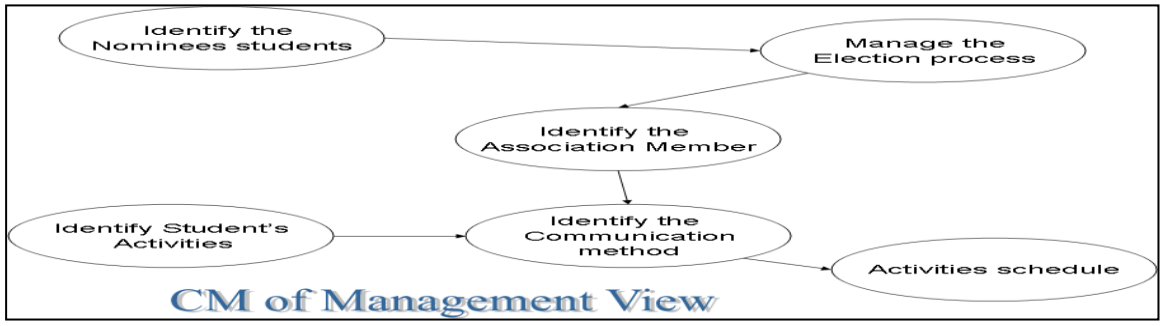


Figure5-12: CM of Management Member View

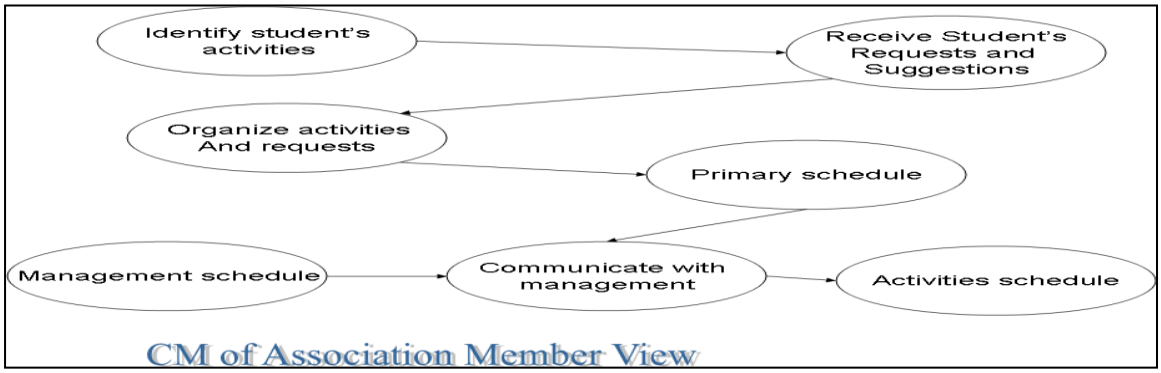


Figure5- 13: CM of Association Member View

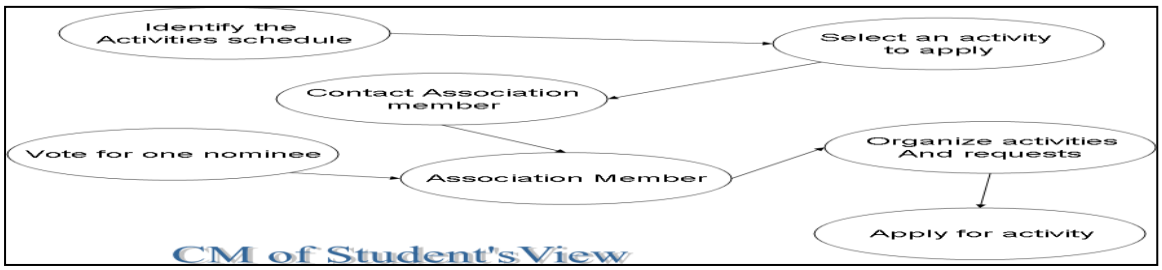


Figure5- 14: CM of Student View

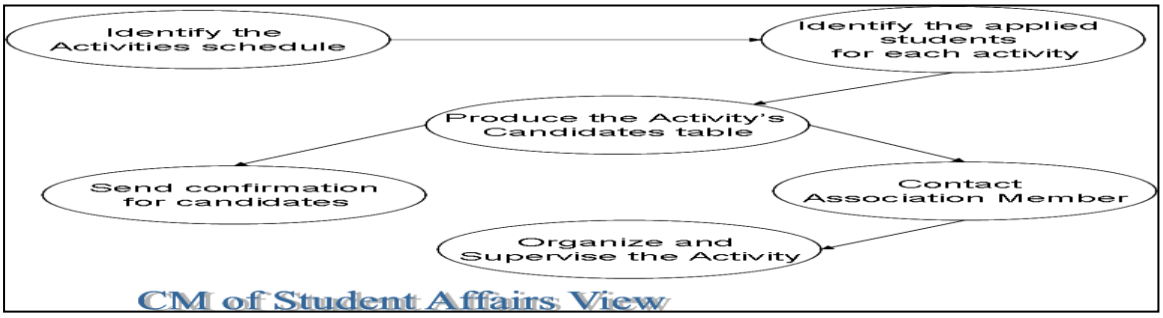


Figure5- 15: CM of Student Affairs View

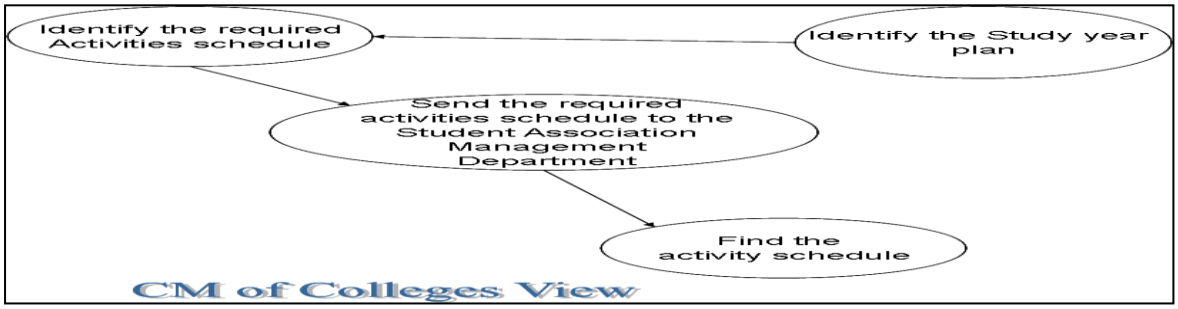


Figure5- 16: CM of Colleges View

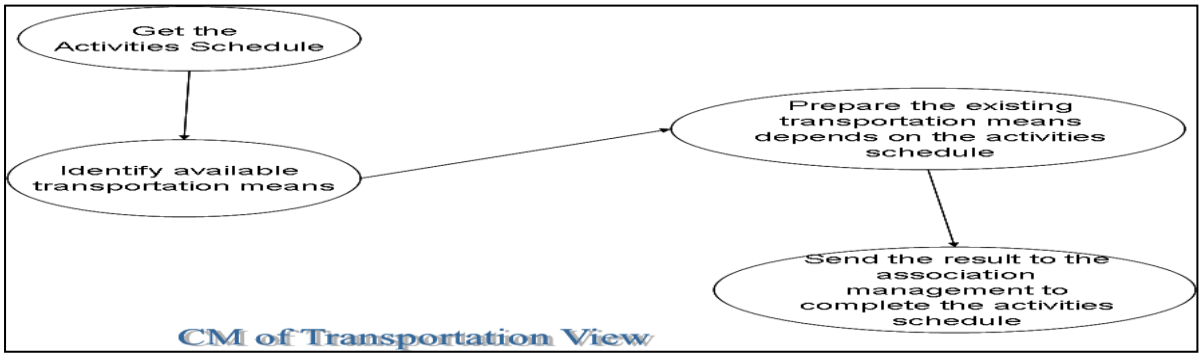


Figure5- 17: CM of Transportation View

The consensus primary task model (CPTM) is derived from the above views and represents all the points agreed by different stakeholders; the CPTM for SAS is presented in Figure 5-18.

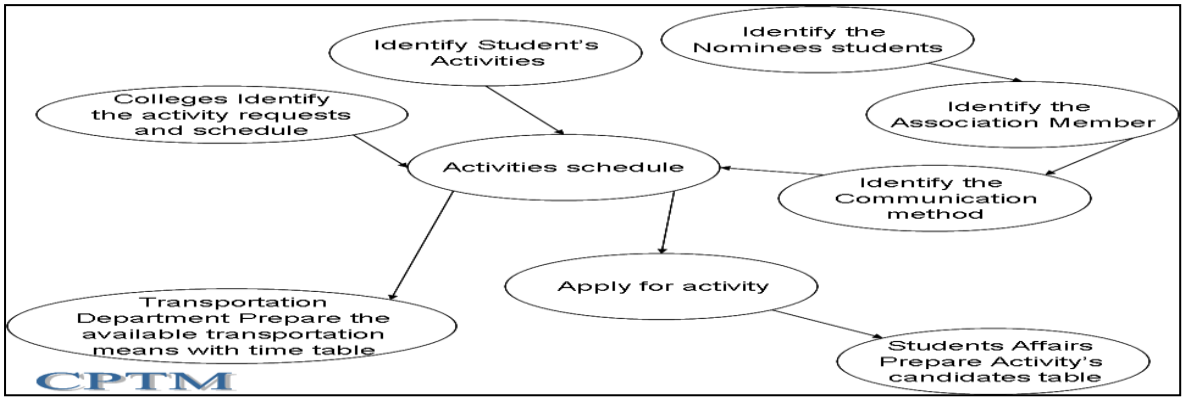


Figure5- 18: The Consensus Primary Task Model (CPTM) of SAS

5.3.2.4 Comparing the conceptual models to the real world

As previously mentioned, the SSDDD framework describes the role of soft system methodology, which requires the investigator to compare the conceptual models with the actual real life situation, and if there is no real world system available, then the developed conceptual model will be used as the real world system model. Here, the developed conceptual models were considered as actual system models, as the SAS available was a manual one and there were no real life models available to compare with the above developed conceptual models. Based on this, the students used the developed conceptual models as a base from which the SAS is modelled as a domain model. The consensus primary task model (CPTM) is developed from these conceptual models, which is further used with SSM to generate the domain model.

5.3.3 Post1-SSM Phase: Moving from Soft language (SSM Phase) to Domain Model

5.3.3.1 Moving from SSM conceptual model to UML use cases

As in the previous case study, UML was adopted here to model the domain model, for which the conceptual model is converted into use cases and use case modelling by using the conversion method explained before,. The extracted use cases are then used to develop a UML sequence diagram, class diagram and activity diagrams. The next subsection will show the conversion from CM to use cases.

1- Use case derivation from conceptual model

The student group used the transition method, which is explained and presented in Figure 4-11, Chapter 4, as part of the SSDDD framework approach, to move from SSM and consensus primary task model (CPTM) to UML use cases. The stage of moving from SSM conceptual models to a use case is eminently difficult, and needs a clear distinction between stakeholder goals, business activities and use cases. The students identified the use cases for SAS and reported that the developed model represented a hierarchy of business activities related to the stakeholder goals, which had encouraged the development of the system. The identified use cases for SAS, together with the embedded activities in each use case (Table 5-7) and the use case diagram, are presented below:

Use case Name	Use Case activities
1- Add a new nominee	Select menu option, enter nominee's details into form, and enter the nominee brief and targets.
2- Collect students' votes and choose the association member	Select menu option, find all nominees and their details, vote for one nominee only, and select the association member
3- Add a new activity	Select menu option, enter activity details into form, enter the limit for the number of students that the activity can handle
4- Generate activities schedule	Select menu option , gather all activities and details entered into an interactive table
5- Apply for activity	Select menu option, view the interactive activities table, select the activity that would like to apply for, check if the activity limit has been reached or not, Confirm if the student is accepted for the selected activity or if the limit has been reached
6- Communicate with association member	Select menu option, view the contact details for the association member, enter student's message into a form with his\her contact details, email the association member with the student message

Table5- 2: SAS use cases

This student group preferred to present the use case activities in the above format rather than utilise a use case proforma format. They clarified at this point that the use case diagram they had prepared was a detailed one, and all the activities required to draw the activity diagram were listed.

2- Generating activity diagrams based on use case diagram

The student group created 6 activity diagrams, which are presented in Appendix 4.

3- Generating sequence diagrams based on use case diagram

The student group went a further step in doing what the framework asked by giving a description of the use cases. They prepared three sequence diagrams. They defined the sequence diagram as a kind of interactive UML diagram that showed the operation of processes among each other along with their order of occurrence. The three sequence diagrams which they prepared are presented in the following figures.

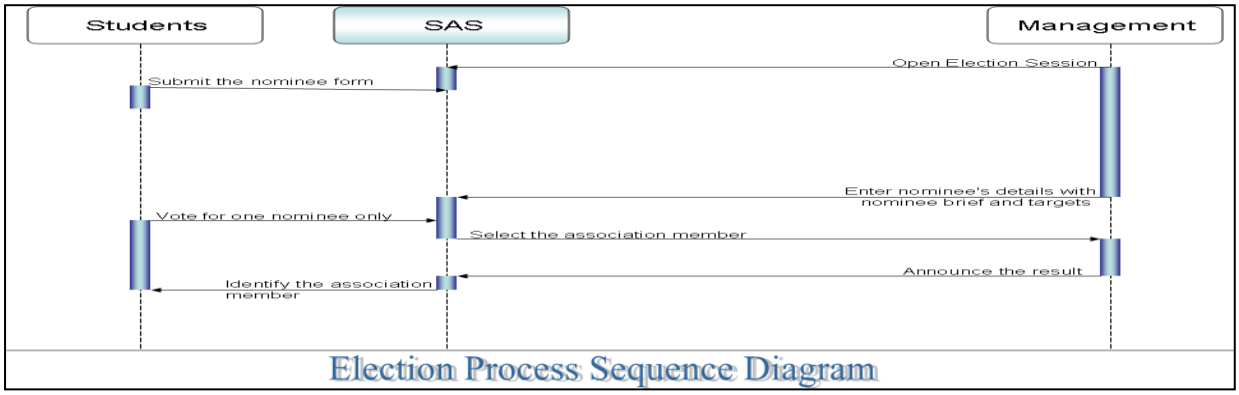


Figure5- 19: Election Process Sequence Diagram

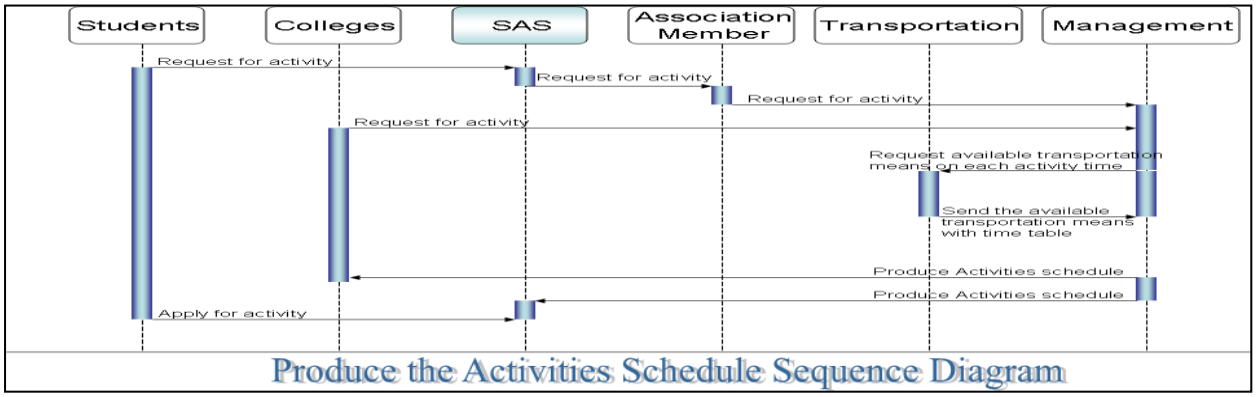


Figure5- 20: Produce Activities Sequence Diagram

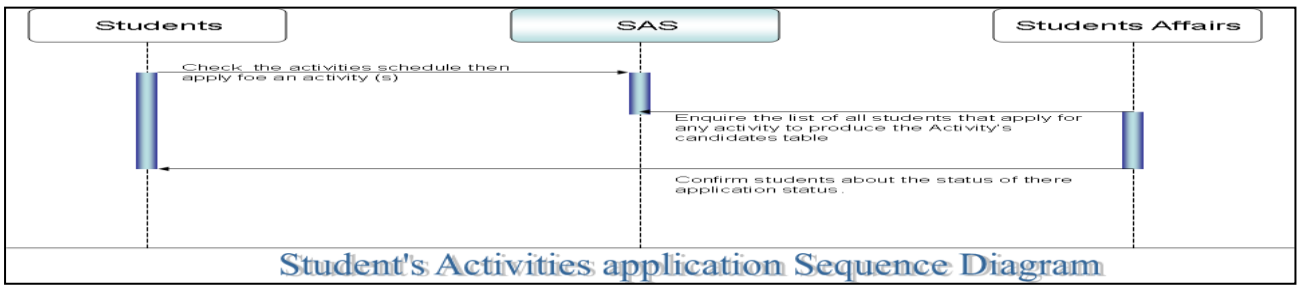


Figure5- 21: Student Activities Application Sequence Diagram

5.3.3.2 Developing the class diagram based on use case and activity diagrams

Lunn (2003, p.19-20) defines a class diagram as "a collection of all classes and the relationship between them", which "defines the static structure of the system". The student group draw the following class diagram to represent SAS:

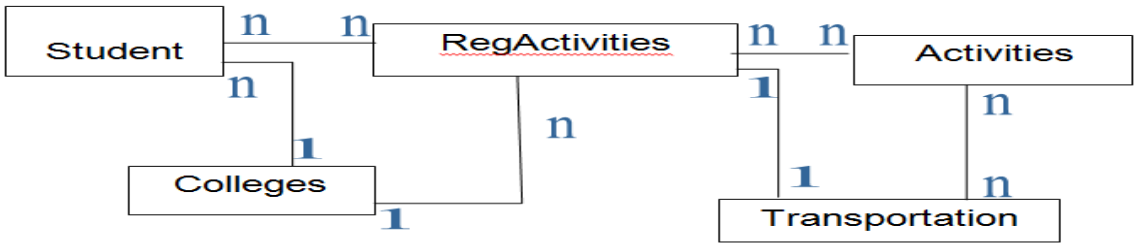


Figure5- 22: Class Diagram of SAS

5.3.4 Post2-SSM Phase: Software Implementation

The Naked Objects implementation pattern was used as recommended by the SSDDD framework. A brief description of the implementation of SAS using Naked Objects and other supported software, as done by the students, is presented in the following sections. An evaluation of the implementation using the implementation pattern is also presented, as well as a reflection on the framework as a development approach. Here is a group of screen shots from the implemented software are presented in Appendix 5.

5.3.4.1 Implemented software evaluation and testing

The students reported that they tested the implemented system based on two factors, the interface factor and the coding factor. For the interface factor, they tested whether or not the system contained interfaces for all the stakeholders; whether or not the interfaces were simple and easy to use; and whether or not the interfaces matched the stakeholders' respective requirements.

With regard to the coding factor, they tested the following issues: reduction of bugs/errors that can be generated from code conflicts and code efficiency (getting the same result within the best time and with the fewest resources). The testing process flowchart is presented in Figure 5-23.

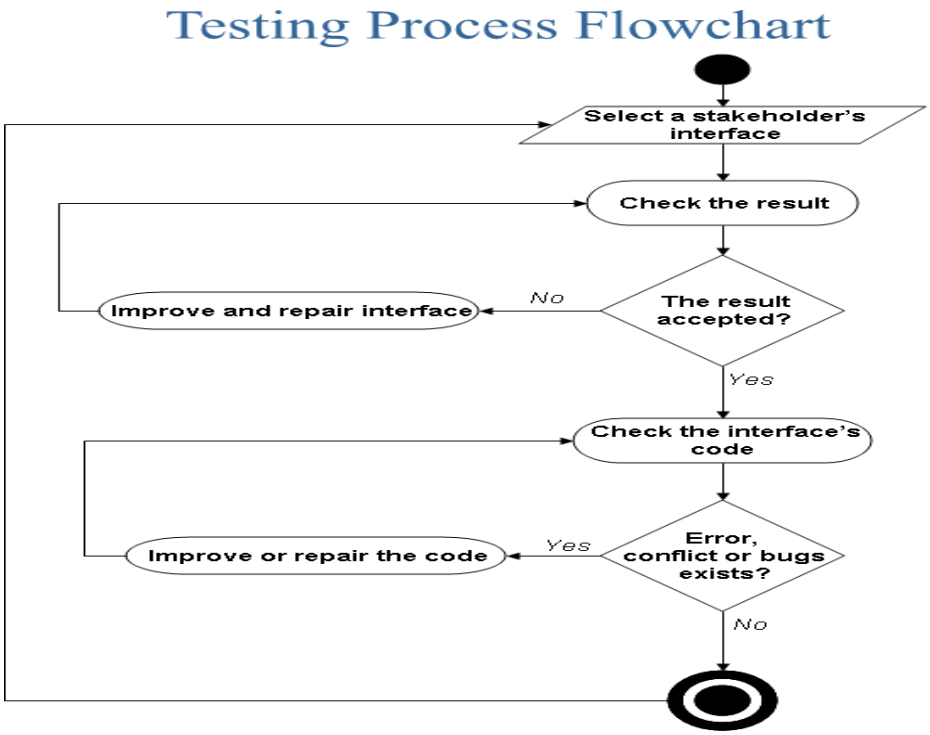


Figure5- 23: Testing Process for SAS

The students revealed that they had followed this testing strategy for ensuring the adequate implementation of the system as per its design, for reflecting on the framework requirements, and for gaining benefits from the learning process by making further changes to enhance the system. They declared that the testing objectives determined were achieved, and the system could be used by the department.

5.3.4.2 Reflection on the SSDDD framework

After developing the system, the students reported the following benefits:

1. The utilization of SSDDD framework helped them to improve their development and documentation skills.

2. The adoption of the framework as an integrated approach for software development was beneficial to comprehend the soft and hard requirements.

However, the students raised certain issues regarding their project, which are summarized below:

3. The time frame allowed to complete this project was not suffient, since the students needed to explore different aspects of Naked Objects, as it was new to them, and required more practice to improve their professional development.

4. The required resources must be available, especially original copies of Naked Objects rather than trial versions. Also, more time is required to deal with Naked Objects, but if given enough time, some of them will handle it well.

5. As the students were junior developers, they insisted that the developed system had a high potential of further enhancements and refinements. They hoped to improve the system so that it could be available online for any member to access remotely.

Therefore, it can be reflected that though the framework provides successful implementation of the system, it needs time to comprehend all the related concepts and gain proficiency. For an undergraduate student, more training is required to understand the SSDDD framework, along with high availability of resources.

5.4 Postgraduate Project: Schools Liaison Coordination System

The methodology Chapter stated that the framework is evaluated as a development approach to an iteration process (action research). First, the undergraduate students applied and evaluated the framework in their projects (these students are considered as junior developers), and their feedback has been used for further development and enhancement of the framework. The next step is to apply and evaluate the use of the framework as a development approach for postgraduate student projects with a different domain. This step will be presented here in relation to the Schools Liaison Coordination System (SLCS) project, where any feedback will enrich the next iteration and be applied to another postgraduate project.

In the summer of 2009, the postgraduate student Saraj Din selected the development of the liaison coordination system within the School of Computing and Engineering as his project. This system utilized the SSDDD framework. The school wanted to develop a database system to replace the existing one based on EXCEL. It was requested that the new system would analyse the data and also compare it against the previous years. It would be required to use the EXCEL reports and generate cumulative reports by grouping them as per the subject areas to provide an analysis of the applications. Also, the system would need to integrate the contacts database for additional information to compare targeted schools year by year.

A description of the SSDDD framework and its application by the undergraduate students were provided to Saraj Din (2009), to assist him in understanding the work. The project

was commenced under the supervision of Dr. Steve Wade and with the current researcher as co-supervisor. Saraj Din (2009) started the work by identifying the aim of this project, which is to design and develop a database-driven reporting system by using SSDDDF to achieve the following objectives listed in table 5-3.

- To consider the newly developed 'Systemic Soft Domain-Driven Design Framework (SSDDDF)' and whether a software application could be developed using this framework.
- To analyse and investigate the client's problem by applying an appropriate methodology such as soft system methodology (SSM).
- To identify user requirements as a set of use cases.
- To build a well-architected software application using a suitable tool that will be acceptable and applicable at the University of Huddersfield.
- To test the built application using a test plan devised from the original set of use cases.
- To evaluate the developed system.
- To ensure the developed system is flexible, secure and scalable, as well as a fully functional to meet the client's requirements.
- To ensure the client will be able to save applicants' (students') data.
- To ensure the client will be able to analyse applicants' data from various UK schools by generating different reports.

Table5- 3: The objectives of database-driven reporting system

After this, he began to apply the framework using the feedback from the undergraduate students' work and the description of the framework given to him. The use of feedback to increase learning is at the heart of the methodology applied to evaluate this framework as an iteration process. The guiding methodology of SSM is a key part of SSDDDF, and the enforced learning which it contributes is a major benefit of using it. The application and evaluation of SSDDDF is presented in the following sections.

5.4.1 Pre-SSM Phase

5.4.1.1 Initial problem identification

Saraj Din (2009) conducted different meetings with the school staff in charge of admission. He identified that under the existing system, the students' applications for admission received at the University Of Huddersfield School Of Computing and Engineering were sent to the Recruitment Coordinator on a monthly basis in the form of an MS-EXCEL report

consisting of hundreds of records with precise information. The task of analysing this data and make comparisons was quite tedious and time consuming. For these reasons, he identified the problem to be: "*To develop a system that takes EXCEL reports to generate cumulative reports to provide analysis of applications by grouping them across subject areas and integrating contacts database for additional information to compare targeted schools year on year".*

5.4.1.2 Stakeholders roles analysis

As explained in the framework, stakeholder roles analysis aims to identify and assess the roles of the key people or institutions, which may affect the success of a project. Saraj Din conducted a meeting with Computing Manager Robin Sissons about the availability of the resources to be used in this project. This was important in enabling him to identify the roles of all the involved stakeholders. R. Thompson from "Mind Tool Club" emphasises the significance of the role of a stakeholder by pointing out that "By engaging the right people in the right way in your project, you can make a big difference to its success".

Thus, for the success of this project, Saraj Din (2009) made it a priority to identify the exact roles of the stakeholders involved in the Schools Liaison Coordination System. He identified the following stakeholders:

- The primary stakeholder is the client, the recruitment coordinator Lorraine Gearing, whose role is both administrator and user of this system.
- The School of Computing and Engineering at the University of Huddersfield is also a stakeholder, and provides the resources such as software and hardware to implement this system.

Based on this, Saraj Din (2009) involved the determined stakeholder (client) in the development of his project through regularly scheduled meetings to ensure its success.

5.4.2 SSM Phase

5.4.2.1 Investigating the problem situation using rich picture

As mentioned in the previous cases, any elements can be included in the rich picture since there are no specific rules for drawing it, but the commonly used elements are the actors of the system.

For investigating the problem situation of the Schools Liaison Coordination System, Saraj Din(2009) came up with the rich picture presented in Figure 5-24.

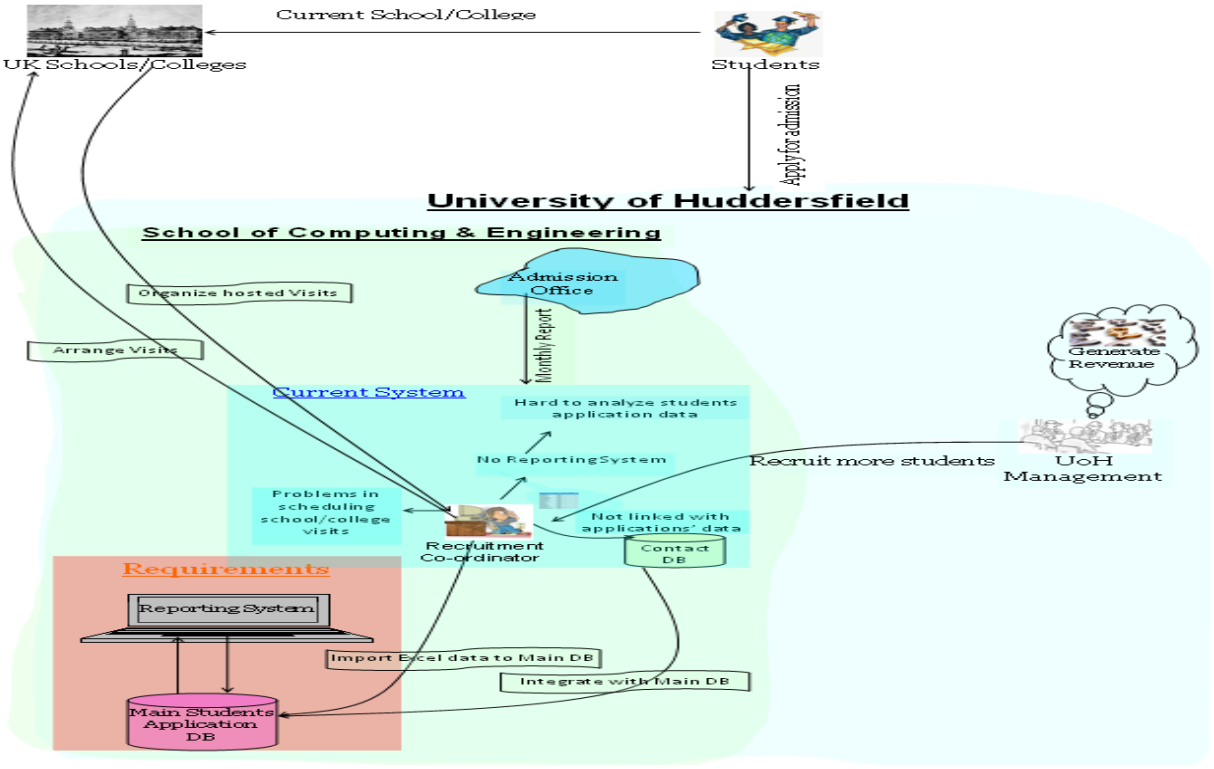


Figure5- 24: Rich Picture of the Schools Liaison Coordination System

5.5.2.2 Modelling the system using root definition

As explained in Chapter 4, root definition is describing the system purpose of the interested stakeholders. According to SSM, the root definition explains the core perception of the system to be modelled. It is then tested using Checkland's mnemonic CATWOE. The root definition is used for constructing a conceptual model (CM) or consensus primary task model (CPTM).

The root definition for the Schools Liaison Coordination System, as identified by Din (2009), is presented as follows:

"A Liaison Coordination System that imports Excel reports, integrate contacts database for additional information to generate cumulative reports to provide analysis of applications of students by grouping them across subject areas, and to compare targeted schools year on year to save time."

5.5.2.3 Modelling the system using the conceptual model

The conceptual model describes the activities that might take place if the relevant root definition is an accurate representation of the system under development. The following conceptual models (CMs) were developed by Saraj Din (2009), based on the previous works mentioned above.

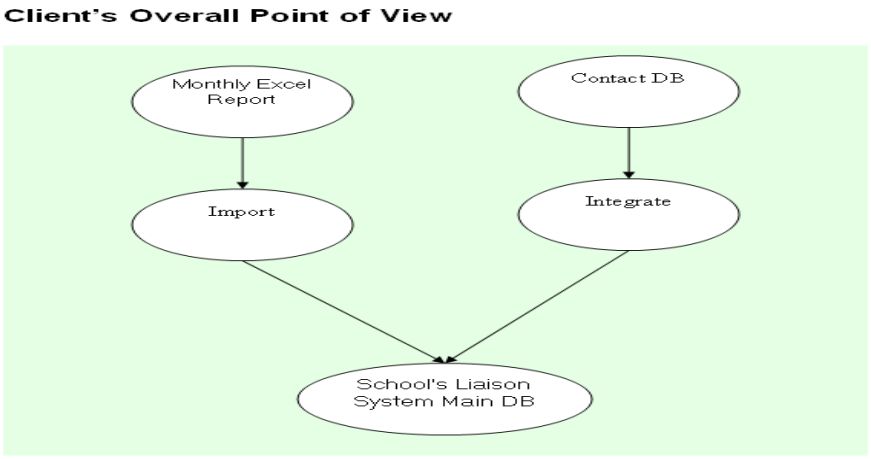


Figure5- 25: Client's Overall Point of View

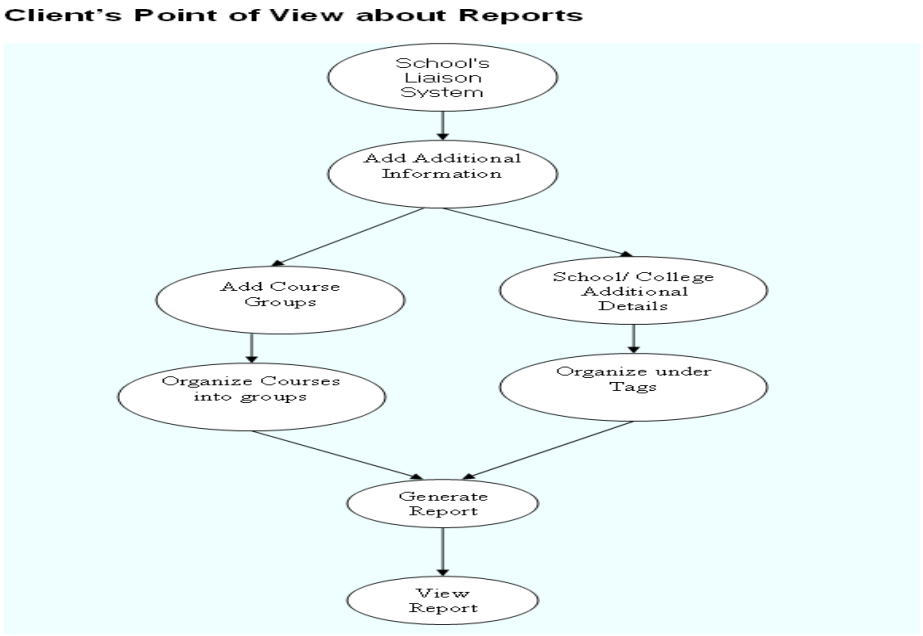


Figure5- 26: Client's Point of View about Reports

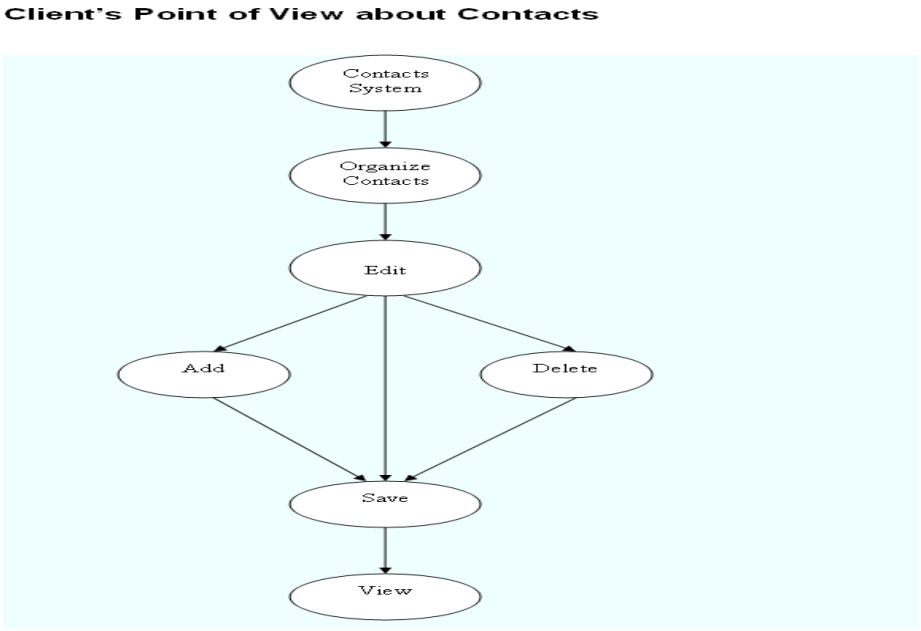


Figure5- 27: Client's Point of View about Contacts

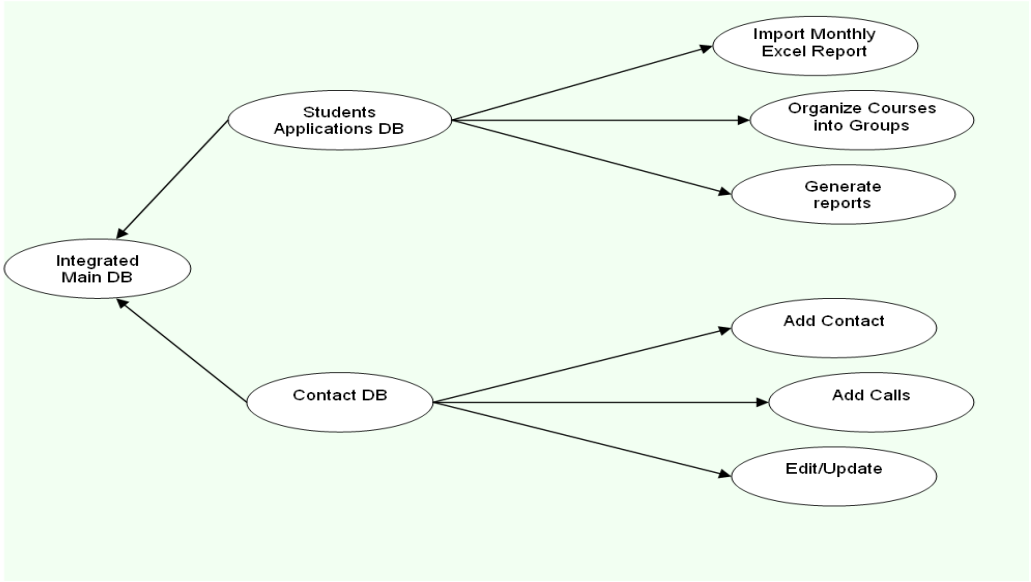


Figure5- 28: Consensus Primary Task Model (CPTM)

5.5.2.4 Comparing the conceptual models to the real world

Saraj Din (2009) mentions that there was no real life schools liaison coordination system available to compare with the above developed conceptual models. This being the case, the conceptual models were used as a base from which the Schools Liaison Coordination System

was modelled as a domain model. A consensus primary task model (CPTM) is the result of combining all the developed conceptual models. The other output models from SSM and the CPTM are the major components of soft language, and these were used to generate the domain model.

5.4.3 Post1-SSM Phase: Moving from Soft Language (SSM Phase) to Domain Model

5.4.3.1 Moving from SSM conceptual model to UML use cases

UML is an important modelling language and was adopted here to model the domain model. For this, the conceptual model is converted into use cases and use case modelling. The extracted use cases are used to develop a UML sequence diagram, class diagram and activity diagrams. The next subsection will show the conversion from CM to use cases.

1- Use case derivation from conceptual model

Din (2009) used the SSDDDF approach to move from consensus primary task model (CPTM), generated through SSM, by converting it into UML use cases. As described earlier, the SSDDD framework adopts the transition method explained in Chapter 4.

Based on the above method, Saraj Din (2009) reported that the developed model represented a hierarchy of business activities related to the stakeholder goals that fuelled the development of the system. The business activities are represented in a hierarchy of conceptual models, with the lowest model containing more primitive, elementary business activities than the higher ones. Each individual business activity is represented in context, in the image of the conceptual model of which it was a part of. Using the above method, the use cases for the Schools Liaison Coordination System were determined as shown in Figure 5-40.

2- Use case proforma

After deriving the use case diagram from the SSM conceptual model, Saraj Din (2009) developed use case proformas to show the details about each use case. Saraj Din reports that a use case proforma must describe the various components which were presented before in table 4-1, chapter4.

The developed proforma tables to represent the Schools Liaison Coordination System are presented in Appendix 6.

3- Generating activity diagrams based on use case diagram

Saraj Din (2009) created activity diagrams for the Schools Liaison Coordination System, and those created for some of the use cases are presented in Figures 5-29, 5-30 and 5-31.

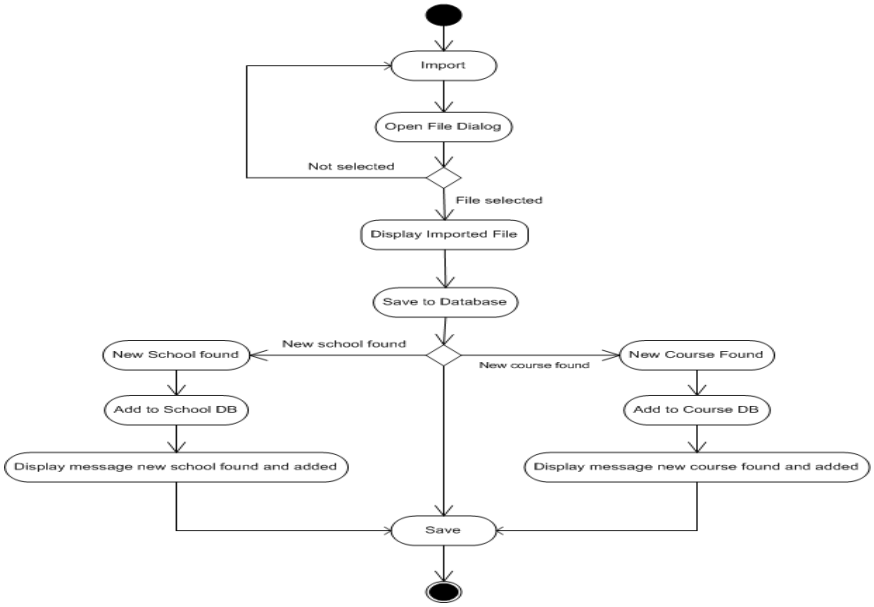


Figure5- 29: Activity Diagram for Import Monthly Report

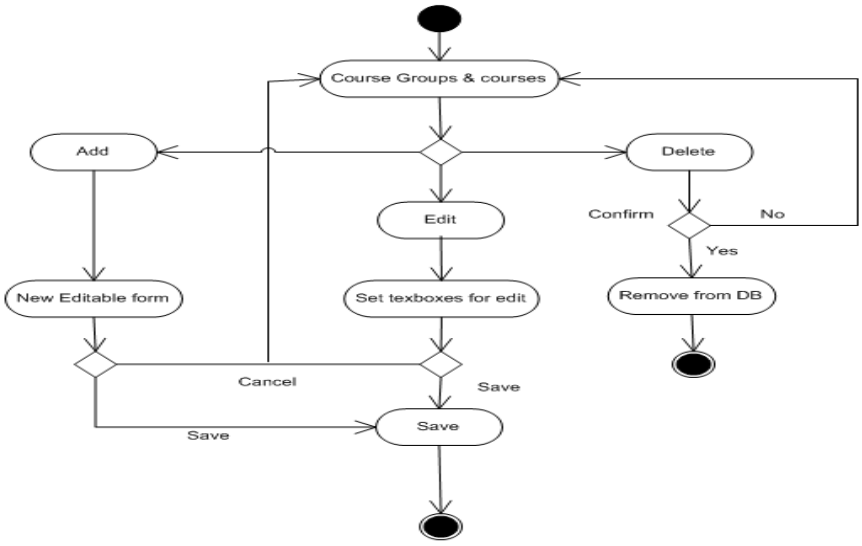


Figure5- 30: Activity Diagram for Add, Edit or Delete Course Groups & Courses

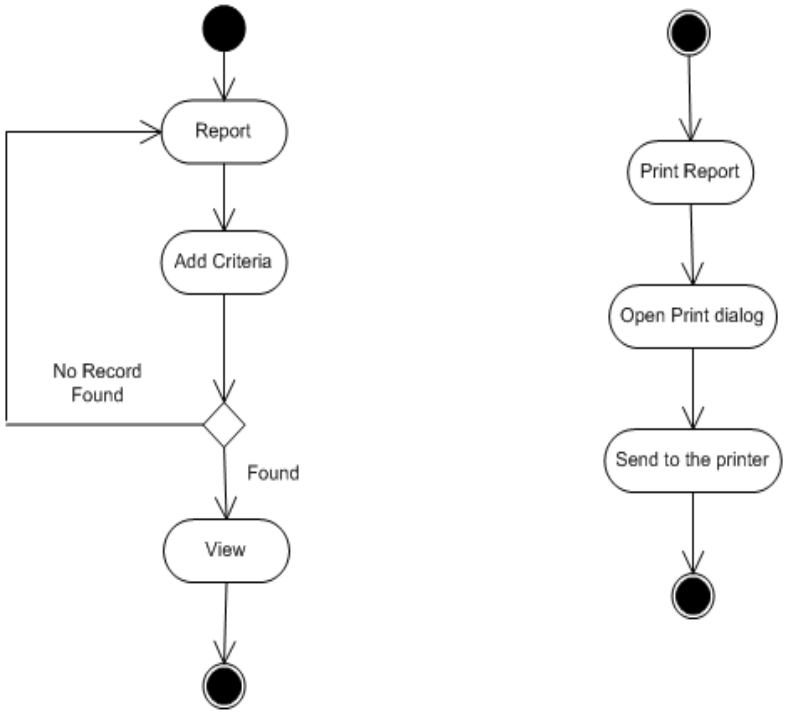


Figure5- 31: Activity Diagram to Generate and Print a Report

5.4.3.2 Developing the class diagram based on use case and activity diagrams

In his project, Saraj Din (2009) referred to Lunn's (2003, p.19-20) definition of a class diagram, which was explained earlier. A class may include a lot of information, including attributes of the data that is to be stored in the system and the operations that could take place. A class diagram is a more detailed representation of a system design. The class diagram is a principle output of object-oriented analysis and design (OOAD). Saraj Din (2009) also identified the three basic types of relationship between classes, the first of which includes one-to-one, one-to-many and many-to-many. The other two types of relationships are inheritance and aggregation, which provides the mechanisms for re-using design and code. Based on the above definition and clarification, Saraj Din prepared the class diagram for the Schools Liaison Coordination System (Figure 5-32), which represents the part of the domain system.

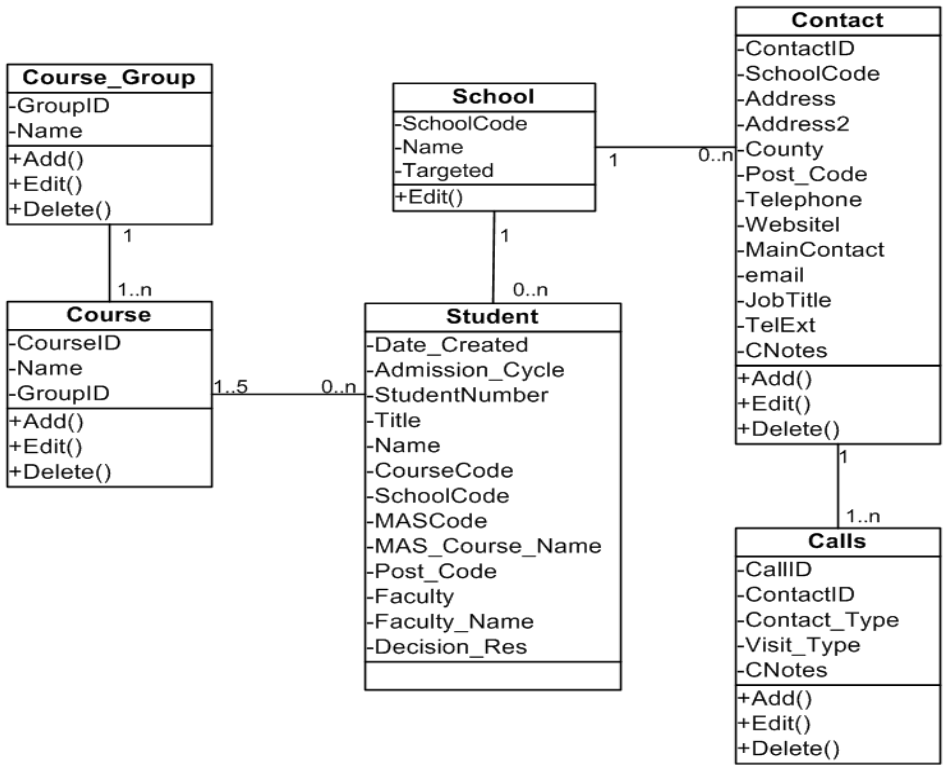


Figure5- 32: Class Diagram of the Schools Liaison Coordination System

In the domain model, the important business logic must be implemented in classes. As an important part of the domain model, the implementation pattern utilize the class diagram to generate the programming code.

5.4.4 Post2-SSM Phase: Software Implementation

Because of certain problems with SSDDDF implementation, Siraj Din (2009) opted to follow an alternative implementation approach. SSDDDF requires specific DDD implementation patterns, such as Naked Objects or alternatively the ADO.NET Entity Framework, but in this project it was difficult for him to apply it due to the following critical issues:

1. The only version of Naked Objects available to him was a beta version that was only applicable for MS Visual Studio 2010, which was also a beta version.

2. In reply to an email from Saraj, Richard Pawson (Managing Director of Naked Objects.org.) explained that the previous version of Microsoft Entity Framework was weak and would no longer be supported.

3. Using ADO.NET Entity Framework was highly time consuming, and as a new developer, the task would be difficult because it has its own new query language (Entity-SQL), which is entirely different from standard query language (SQL).

4. Entity-SQL does not support DML statements (insert, update, delete) and also some other programming requirements, and without DML he would be unable to develop an import wizard.

Based on the above, Saraj Din preferred to continue with the traditional object-oriented approach to design the system structure and database, and then proceed to the implementation process. He decided to use visual basic as an implementation language and SQL Server 2008 as a database server. He argued that the Microsoft.net framework provides full support for multiple tier applications, whereby different layers can be easily managed into separate components using built-in classes.

5.4.4.1 Implementation evaluation

After examining the above problems, it is clear that none of them are directly related to the use of SSDDDF in the implementation process. All of them are related to the availability of resources and the time required by the developer to adopt new information system development approaches. Because of the time constraint, another implementation approach was selected and used. This situation helped to raise awareness that with the next case study, all the necessary resources must be available, as well as the skills required to deal with the implementation patterns. This issue will be clarified in the next section, which deals with the use and evaluation of the SSDDD framework in the other postgraduate student projects.

5.4.4.2 Reflection on the SSDDD framework

The postgraduate student Saraj Din (2009) explains that the purpose of using SSDDDF was to discover if he could use it to develop a software application. In his evaluation, Saraj Din (2009) mentions the benefits of SSDDD framework, which are presented as follows:

1. SSDDDF enables the researcher to understand and explore the problem situation better through SSM. It enables the comprehension of different views of the current situation through the stakeholder analysis and root definition modelling stages. This can facilitate an understanding of the business objectives and how activities are done.

2. It enables the developer to build a better application that suits the users' requirements, and even to build a system that improves on those requirements. The UML stage helps the user to model the system well and to understand the system requirements exactly.

However, he adds that it was difficult for him to use Naked Objects because of the unavailability of resources, and he was also not prepared to implement the software using the Naked Objects implementation pattern.

Looking at the above mentioned problems, it is evident that they are not related to the nature of the framework, but to the developer himself. Such problems can be solved before starting any project by ensuring that the developers are ready to use the framework completely, not partially as happened with Saraj Din. On the other hand, this point can also be regarded as a positive outcome, as it ensures the high compatibility of the framework with the use of other tools for implementation. Sairaj Din used the framework to investigate and model the system, and when it came to the implementation, he used other tools which were compatible with the framework.

5.5 Postgraduate Project: Peer-Tutoring System Development

In the summer of 2010, the postgraduate student Joseph Ucizi Mtenje selected the PTS project and decided to use the SSDDD framework to build it, in order to evaluate the framework as a system development approach. A description of the peer-tutoring case study, the framework, and an explanation of how the undergraduate students had applied the module were all provided to him for providing better understanding. The work was commenced under the supervision of Dr. Steve Wade and the current researcher as co-supervisor to guide the student and collect feedback about the framework's application. The implementation part of this project aimed to build an application that would be used to manage the PTS by letting students book the tutoring sessions. It also aims to allow the lecturers in selecting the tutors and tutees on the basis of students' results from the previous year, previous semester or Blackboard quizzes. The tutors would be the students in their final year with good grades, while the tutees would be the students of first or second year, who needed support to improve their skills. The lecturers would be able to load room availability, enhance the booking process, and monitor the progress of the system by monitoring whether the pass rate had increased as compared to the previous year (without PTS). The passing marks, to determine whether a student qualifies for the tutor position or

not, would be determined by the management and set as a business rule. The information system developed aimed to help the administrator of the PTS by enabling the following functions presented in table 5-4.

Actors	Functions can be perform
Tutors	booking a session by themselves, checking their rewards, update their timetable to help tutees to be sure from tutors availability before booking
Tutees	booking a session by themselves and sign attendance to help the management and lecturers to monitor the progressing of the system
Lecturers	can insert all information about rooms, tutors and tutees. Lecturers will also be able to calculate rewards due to a tutor based on how many sessions did they run. If any system failure occurs, the lecturers will report to the technical engineers to attend and fix the problem
Management	will be able to see the rewards allocated to a tutor by a lecturer, in order for these rewards to be redeemed. Policies and procedures will be applied to the PTS by the management

Table5- 4: PTS actors and functions

A detailed description of the application of the framework by the postgraduate student Joseph Ucizi Mtenje (2010) is presented in the following sub-sections, which illustrates the usage of framework in developing the PTS.

5.5.1 Pre-SSM Phase

5.5.1.1 The problem identification

The postgraduate student Joseph Ucizi Mtenje (2010) refers to the previous work of Salahat et al. (2009), which reports that both the Department of Informatics in the School of Computing and Engineering at the University of Huddersfield in the UK, and the Information Technology College at Ajman University of Science and Technology in the UAE, offer introductory programming modules for their first year computing students. These modules focus on Java programming. Lecturers faced certain difficulties pertaining to students' understanding of the subject as it required problem-solving skills. Students required more tutoring and practical sessions to help them practice different exercises and thus enhance their understanding and practical skills. Both universities expected that by implementing a peer-tutoring system, the failure rate would be reduced. The departments wanted to identify knowledgeable tutors from the other students and find a means to reward them.

The exact problem was identified by working with the students and interviewing them about the difficulties.

Based on the previous work that had been completed, which included interviewing students and administrators in the departments, new interviews were conducted by Joseph Ucizi Mtenje (2010) with those studying programming modules in the Informatics Department at the University of Huddersfield, as these would be the people using the system. He also benefitted from interviews conducted with undergraduate students in the IT College in Ajman University, UAE, which are reported in the previous work mentioned above. In addition, he received feedback from both the current researcher and Steve, the supervisors located in each of the universities, to clarify certain points about the system. Joseph Ucizi Mtenje also interviewed some of the staff members in the School of Computing and Engineering's Department of Informatics who would use the system in the department. One of them is a lecturer who teaches a programming module in the department and stated that "using PTS for the 'Introduction to Programming' module would help the students to increase their confidence in the class, which will help them to be more creative". The lecturer mentioned that the system must get the results of the students from the database and select those students who achieved higher marks, in order to select them as tutors. The tutors requested to insert their time availability into the system. The system must select those students with low grades to be tutees.

Joseph Ucizi Mtenje also conducted several meetings with the current researcher as a co-supervisor and client of this system. The current researcher was expecting PTS to improve the pass rate and hence reduce the failure rate while decreasing the workload of the lecturer. Also, the training sessions are important for the tutors, as they ensure the consistency and quality of the system. Once the students were comfortable with tutoring sessions, they could start studying immediately, and not wait for the students who were uncomfortable.

Joseph conducted another meeting with the management and administration staff. They stated that they need a system for improving the pass rate to help the university and enhance its reputation. Also, they need the system to be easily managed and operated with low financial expenses. The PTS system will be applied and used within the university rules and regulations.

Finally, Joseph conducted a series of interviews with the students. Since, he had the feedback of interviews conducted in previous work, he tried to determine further issues related to the problems of the programming module. Many of students mentioned that they will be happy to learn from each other rather than from their lecturers who teach formally. According to the students who will be tutees, they are looking for extra skills and knowledge to support them to get higher grades, while the students who will be tutors are looking for some extra money to contribute to their expenses. Also, the students focus on some administration issues such as their difficulty in travelling to other campuses to attend tutoring sessions, and preference to be tutored after 5 to avoid any clashes with their classes. The final point they highlighted was that they preferred the system to be online for allowing them to study from home or anywhere else.

As action researchers, Salahat et al. (2009) conducted the face-to-face interviews informally, so that the participants would feel comfortable as they could see who was interviewing them, and to allow them to express their ideas and suggestions comfortably. The participants were able to explain some ideas through face-to-face interviews in a better manner, for example by using gestures and facial expressions, which may not be fully explained in writing or over the phone. These actions were noted and appreciated throughout the interviews, which would not have been feasible over the phone.

As explained above, similar data collection methods were used to conduct new interviews for collecting different types of data through different types of questions. Joseph Ucizi Mtenje asked questions such as:

1. What is the current system offering?
2. How far can their lecturers go in supporting them with their work outside classroom hours?
3. Would more support in their work outside their lecture hours increase their level of comfort with the module while increasing their skills?
4. What do they think of the idea of PTS?
5. Would they understand better if they were learning from a fellow student who had achieved outstanding grades in the previous year, and they could learn from their experience and achievements?

This reinvestigation and refinement of previous findings is related to the heart of the framework, which, as a multimethodology, has adopted SSM as a learning method along with all the other components embedded in it. In this phase, the problematic situation was investigated comprehensively to enable more clarity, which would in turn support the later stages.

5.5.1.2 Stakeholders determination

The stakeholders in this case may be defined as the people who will be using the system and also benefit from it (Joseph Ucizi Mtenje, 2010). The stakeholders of the required PTS system were determined to be peer tutors, peer tutees, lecturers and management. Stakeholders often have different expectations of a system. The different stakeholders of this system expected that they could achieve the following from using PTS:

- Peer tutors are generally looking for teaching experience to be added to their CVs.

- Peer tutees are looking for extra help.

- Lecturers are looking to reduce their workload, and to determine which students most require tutoring sessions.

- Management seeks to reduce the number of failures on programming modules.

5.5.2 SSM Phase

5.5.2.1 Investigating the problem situation using rich picture

In the investigation carried out by Joseph Ucizi Mtenje (2010), rich pictures were used as a tool to express the views of stakeholders and their expectations from the system being developed. In order to redevelop a rich picture of the situation under investigation, he used a number of information sources to capture views about the introductory programming unit from students, lecturers, the management of the School of Computing and Engineering, and the perspectives discovered in previous cases. Interviews with the school administration and groups of students were conducted to understand the problematic situation of teaching the introductory programming module, and suggestions for solving the problems were set out. The following figure (5-33) represents the rich picture of PTS as drawn by Joseph Ucizi Mtenje (2010), based on the previous work by Salahat et al. (2009) and the new data

collected as mentioned in section 5.5.1.1 'The problem identification'.

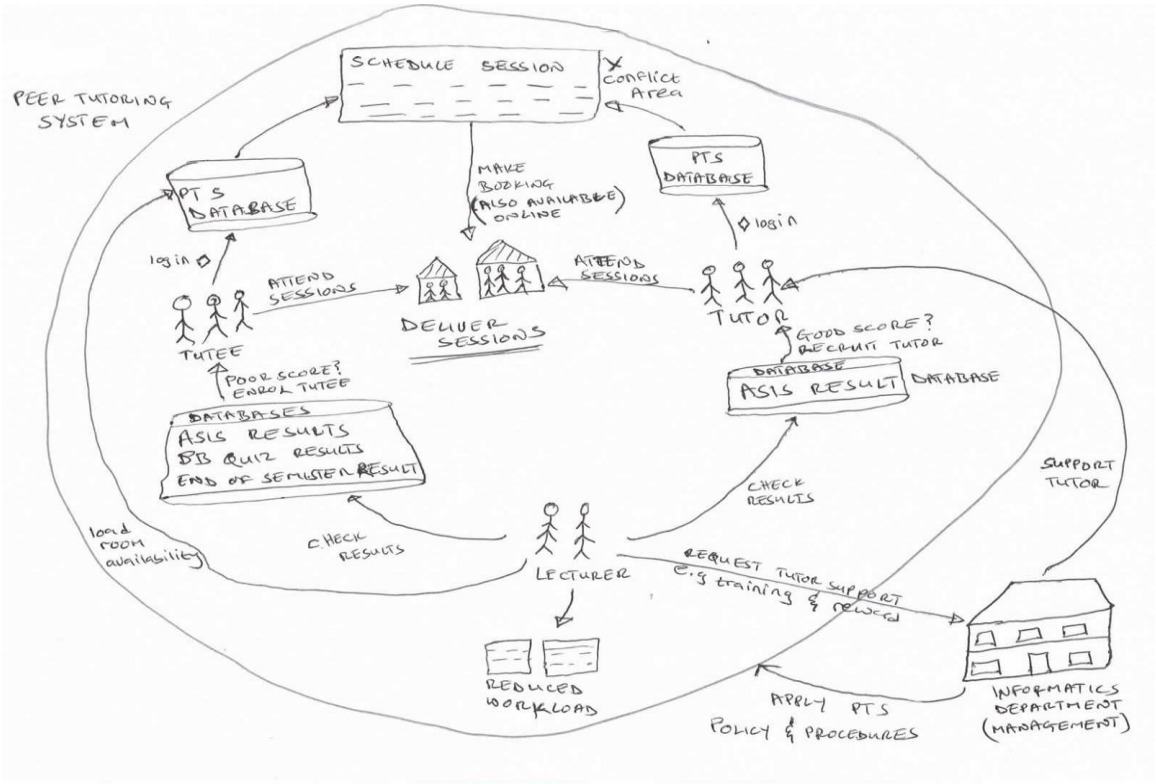


Figure5- 33: Rich picture of the PTS

5.5.2.2 Modelling the system using root definition

In addition to the work previously mentioned, Joseph Ucizi Mtenje adopted Checkland's CATWOE mnemonic and applied it to PTS. He mentions that this transformation is carried out for students, and in this case the students were the customers controlled by actors (the researcher and supervisor). The system activities are controlled by an owner (client), and are performed in a university environment which has established conditions and policies. Using PTS, Joseph Ucizi Mtenje determined the components of CATWOE presented in table 5-4.

CATWOE Component	Description
C-Customers	tutors and tutees whom will benefit from PTS system
A – Actors	Joseph Ucizi Mtenje and supervisors
T – Transformation	The PTS system and the software application to manage it
O – Owners	the client
E – Environment	the university and the its rules and regulations
W-Weltanschauung (world view)	the perception derived from the root definition. This may be views about whether the change is worth doing or not. In this project it is the users' views of tutors, tutees, lecturers, and management about different issues of PTS

Table5- 4: CATWOE of PTS

The root definition for PTS was determined as a compromise between the previous work and as that conducted by Joseph. It is given below:

"To propose a peer tutoring system to improve the pass rate for students studying the undergraduate programming modules in the Informatics Department at the University of Huddersfield, and also to help in the selection of peer-tutees and peer-tutors; the scheduling of tutoring sessions based on the availability of rooms; selection of tutors and tutees; monitoring of perceived benefit to tutors and the progress of tutees in increased self-confidence. Also, the aim is to measure the impact on failure rates and allow the users access to the application to book and deliver sessions without the help of lecturers".

5.5.2.3 Modelling the system using conceptual models

The conceptual model describes the activities that might take place if the relevant root definition is an accurate representation of the working of a system. The following conceptual models (CMs) were developed by Joseph Ucizi Mtenje (2010), based on the previous works and the new data that he had collected. They represent different stakeholders' views, the actions that must be taken based on their views, and also the need to meet the particular cultural, political and social requirements of the system. All of these issues are expressed in the rich picture and modelled using the following conceptual models.

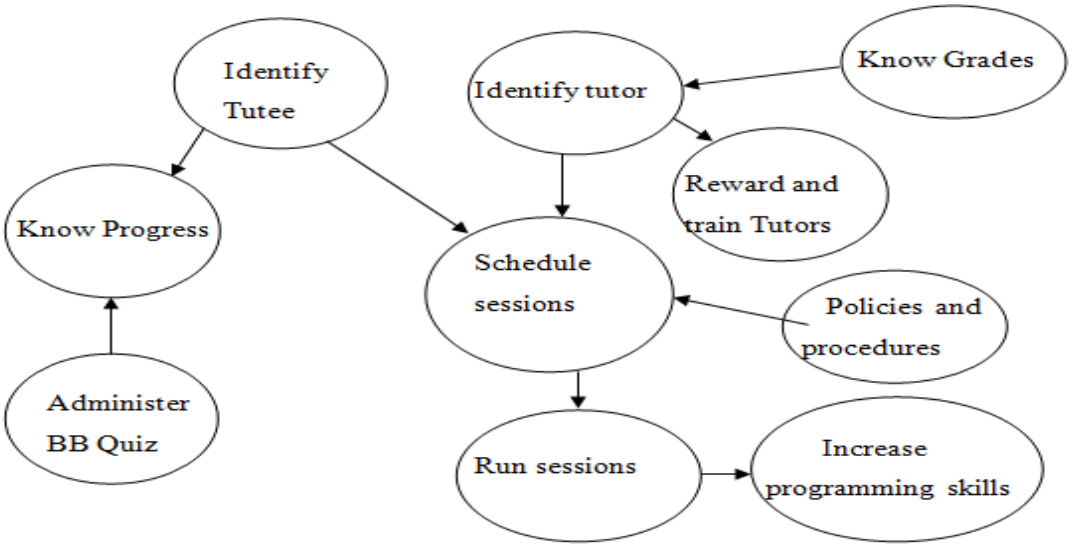


Figure5- 34: CM of Management's View

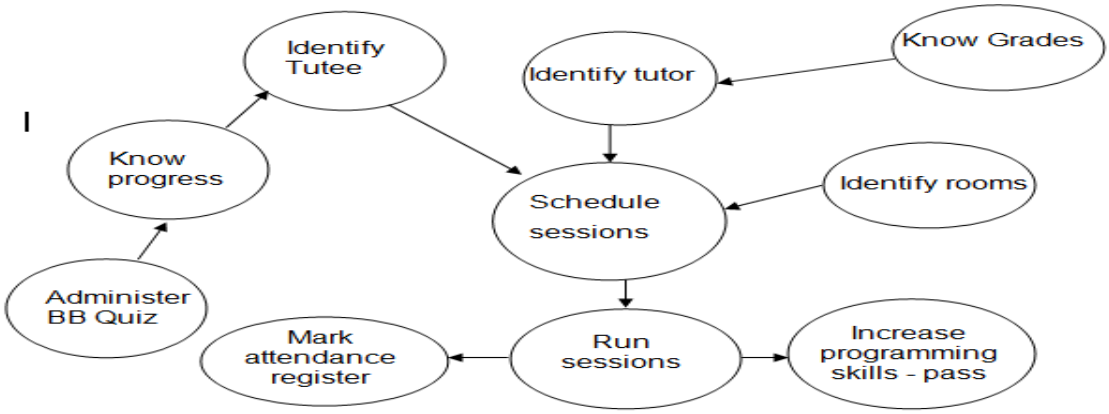


Figure5- 35: CM of Tutee's Point of View

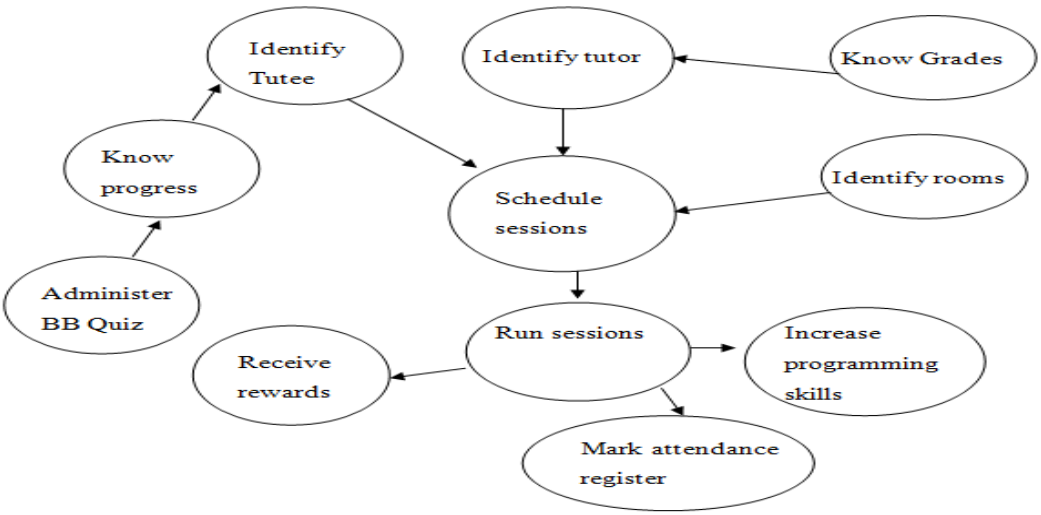


Figure5- 36: CM of Tutor's Point of View

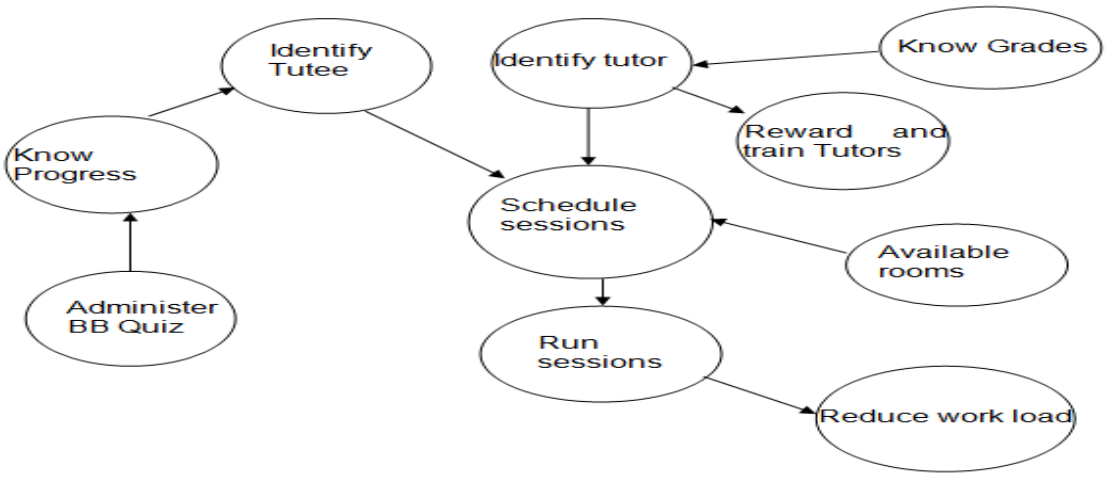


Figure5- 37: CM of Lecturer's Point of View

The above diagrams represent the different views and perceptions of the stakeholders. The proven issues between the different stakeholders are presented in a diagram called the consensus primary task model (CPTM), which represents those points which are agreed by all the stakeholders.

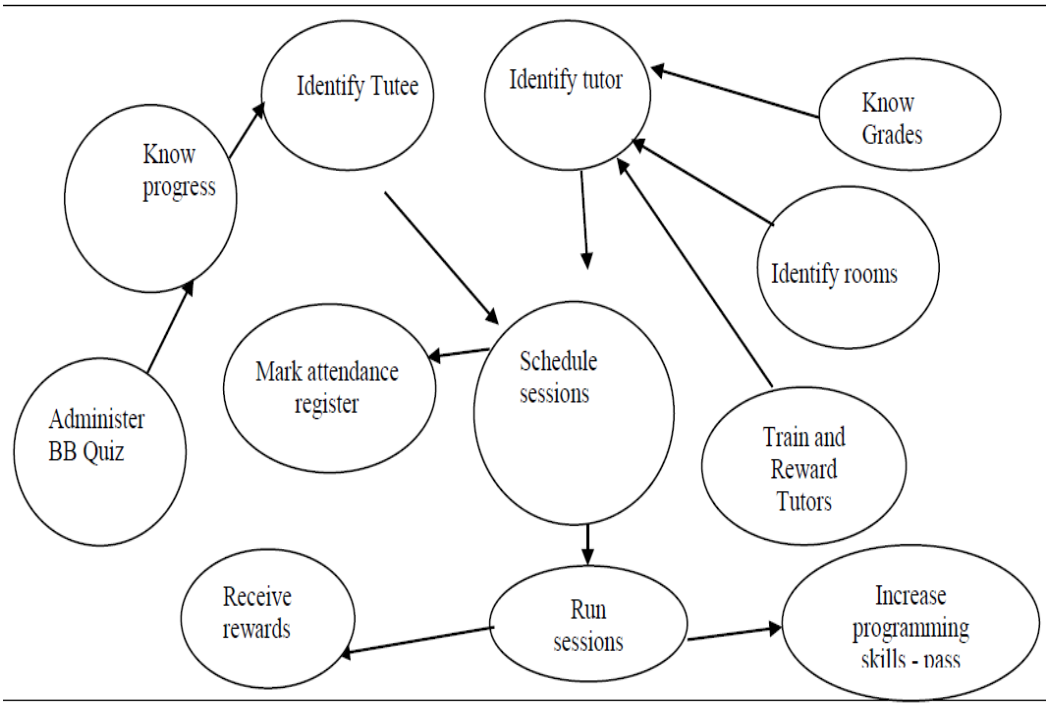


Figure5- 38: CPTM of PTS

5.5.2.4 Comparing the conceptual model to the real world

SSM requires the investigator to compare the produced conceptual model with the actual real life system model. In this project, since there was no existing peer tutoring system, it would need to be critiqued by discussion and making comparisons with another department offering PTS. For example, it would be necessary to consider the internet programming modules which would support this or another university's system if they had one. Also, the conceptual model would be considered as the base to model the PTS system as a domain model. The CPTM, as a combination of all the conceptual models, and the other components of SL will be considered and used in the next phase to generate the domain model, as stated in the earlier stages of the framework.

5.5.3 Post1-SSM Phase: Moving from Soft Language (SSM Phase) to Domain Model

The domain model is represented using UML, which converts the conceptual model into use cases and use case modelling. The next subsection will show the conversion from CM to use cases.

5.5.3.1 Moving from SSM conceptual model to UML use cases

1- Use case derivation from conceptual model

A use case can be represented as a diagram called a use case diagram or through a textual format called a use case proforma. A use case diagram is made up of three key elements, which are actors, use cases and the relationship between them. An actor may be a user (person or thing) of the system or another system, while a relationship is a link between actors who use 'use cases', and sometimes a 'use case' may use another use case or actor. As in the previous work, Joseph Ucizi Mtenje adopted the approach explained in Chapter 4 for conversion from SSM conceptual model to use case model. At this phase, the CPTM models from SSM are converted to UML use cases so that they can be used in the next stage of implementing the application using DDD implementation pattern. The conversion process, as part of SSDDDF, is explained in Chapter 4 and presented in Figure 4-11. Any activity requiring information system is selected as a use case. The stage of moving from an SSM conceptual model to a use case is not as straightforward as this discussion would suggest. In thinking this through, it has proved necessary to make a clear distinction between stakeholder goals, business activities and use cases. The Conscious Primary Task Model (CPTM), which is generated through combining SSM conceptual models, is used to map the activities to use case diagram using the elaboration technique, and stated that use cases are used to model the business domain activities based on DDD concepts.

Joseph Ucizi Mtenje (2010) cited Salahat et al. (2009) and illustrated that when SSDDDF is moving through the process of converting from SSM soft languages to UML diagrams, it requires mapping of the activities from SSM conceptual models, only after a proper understanding of the user requirements and problem situation, to use case diagrams that represent the functionality of the proposed system while still maintaining the user requirements and business activities from the conceptual models in a one-to-one relationship. This will result in some conceptual models being combined and others being decomposed. The use case diagram provides a hierarchy of business activities concerning with the goals for stakeholders that led to the need of developing a system as it is defined in the problem definition in the SSM stage. The conceptual models are arranged in a hierarchy in which the more primitive and elementary business activities are lower than the others. An image of the conceptual model will represent an individual business activity of that part. Using the above conversion algorithm, the conceptual model of PTS presented above is converted into different use cases. The following use case diagram (Figure 5-39) (Joseph Ucizi Mtenje, 2010) presents the result of the conversion process.

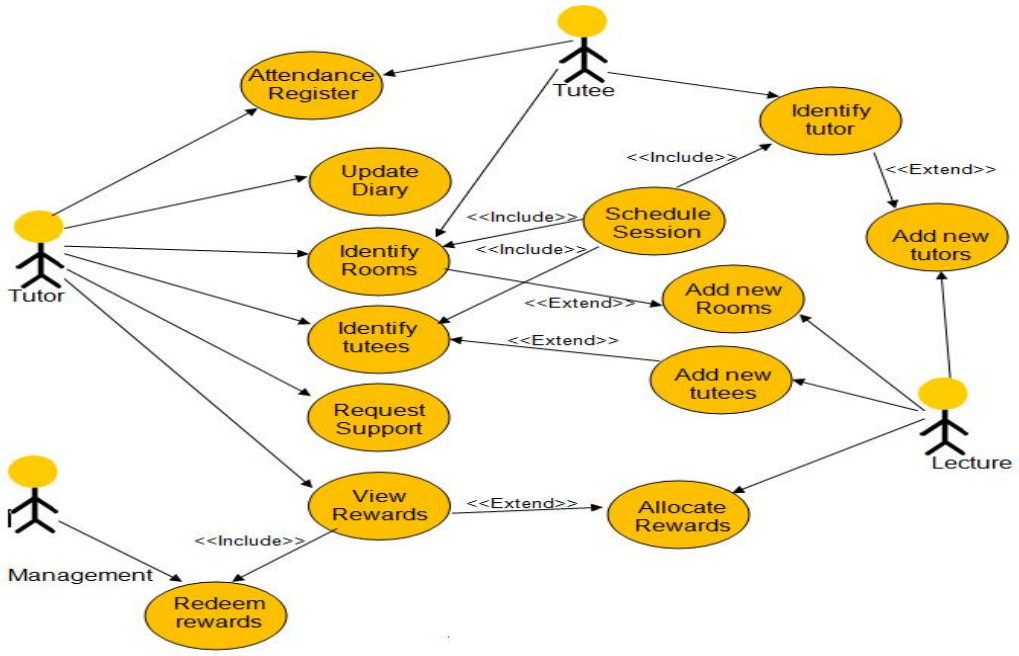


Figure5- 39: Use Case Diagram of PTS

2- Use case proforma

After derivation of the use case diagram from the SSM conceptual model, the use case proforma is prepared to show the details about each use case. Joseph Ucizi Mtenje (2010) developed the use case proformas for the PTS system, which are presented in Appendix 7.

5.5.3.2 Generating activity diagrams based on use case diagram

Activity diagrams are a part of the domain model being is used to implement the information system. Activity diagrams present the stepwise stages of the business process or the software process from starting point to the end; this process may be carried out by people, software components or computers. Each diagram shows the activities embedded in any use case within the use case diagram representing the system.

Activity diagrams will be a part of the domain model used to implement the PTS system as an information system.

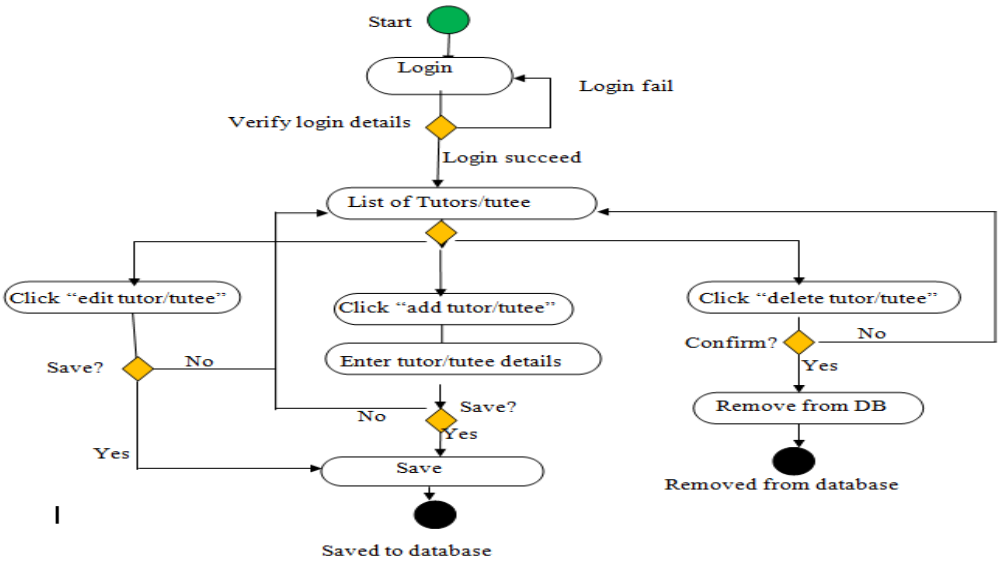


Figure5- 40: Activity Diagram to Update a Tutor or Tutee

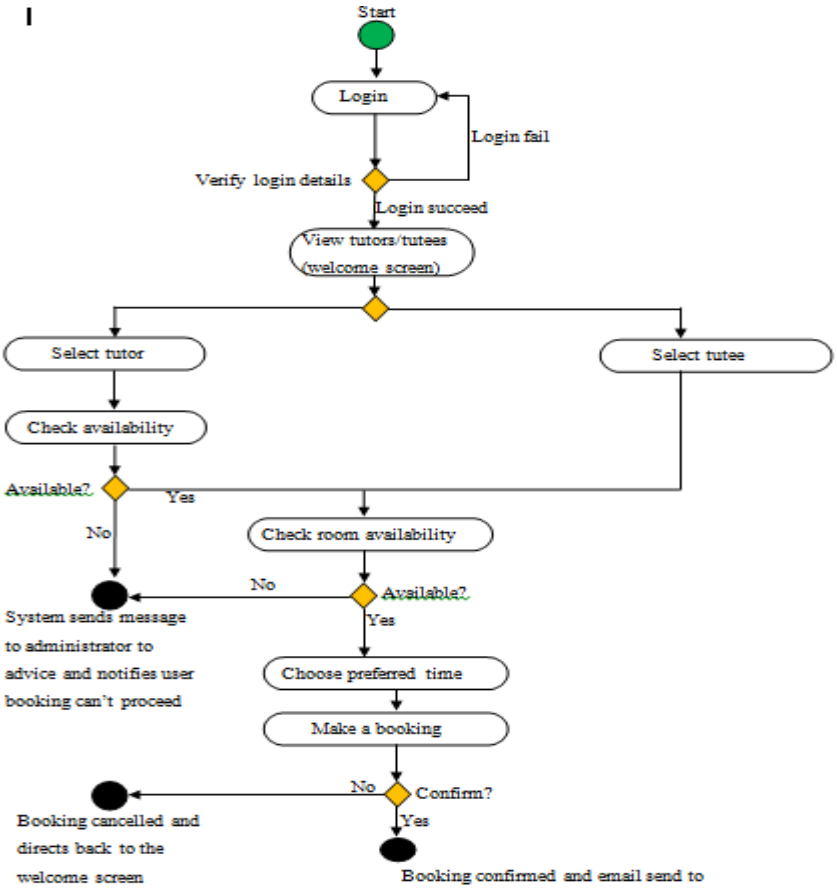


Figure5- 41: Activity Diagram for Scheduling a Session

5.5.3.3 Generating the class diagram based on use case and activity diagrams

A class diagram is a representation of the basic structure of a system. It shows the presentation of the classes in the system, the linkage between them and the number of links. It is a more detailed presentation of the system (Oliver & Kent, 2009). Each use case is presented using a textual template, activity diagram and sequence diagram, and all of them are combined in a use case diagram. The next step in the process is to take the business logic identified in the use cases and associate it with the classes in a class diagram. Following the guideline that all important business logic must be implemented in classes of the domain model, it is used to generate the programming code through the implementation pattern. The class diagram of PTS is presented in Figure 5-42 (Joseph Ucizi Mtenje, 2010).

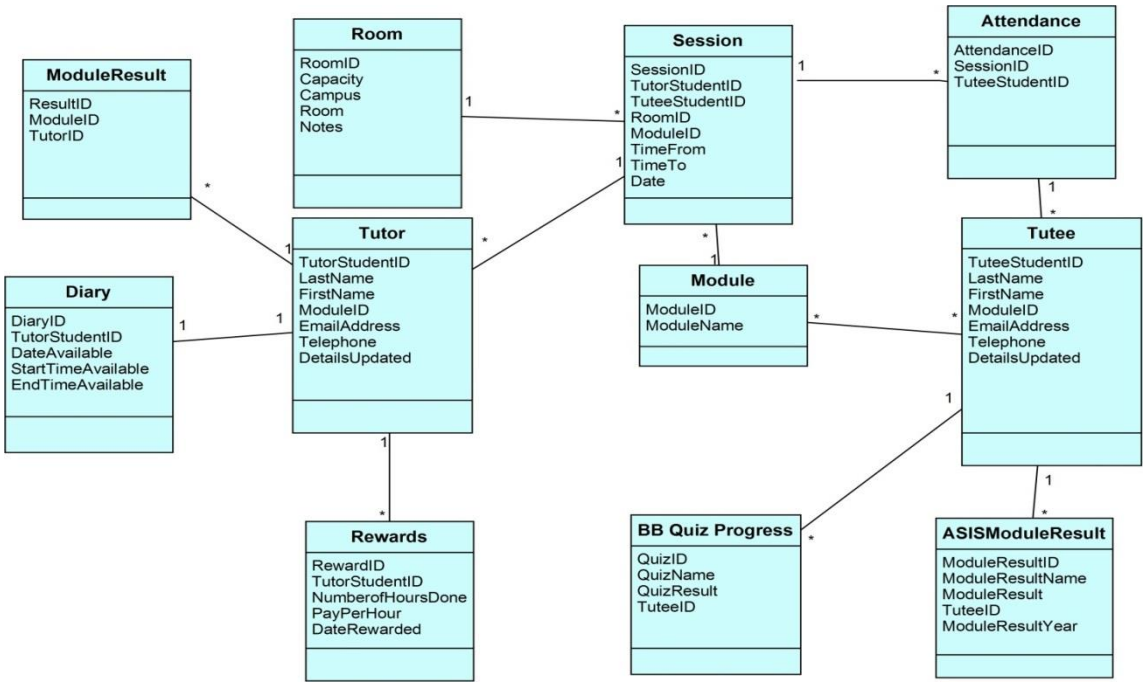


Figure5- 42: Class Diagram

5.5.3.4 Change report generation and refinement

As shown in Figure 4-1, which represents the SSDDD framework, a reconsideration of previous stages is required to refine what has been done during Pre-SSM, SSM and Post1-SSM. This refinement is essential to be sure that the exact changes required have already

been modelled well as a domain model. As a guiding methodology, SSM focuses on the generation of the required change report for the system to be recommended to manage the action (Checkland & Poulter, 2006; Checkland, 1999; Checkland & Howell, 1998). Therefore, before leaving this stage, the domain model should be refined and made ready for implementation.

At this point, the methodology is completed and can be restarted again if any further improvement of the situation is required. It was at this point, where PTS and its application could be implemented and used to serve the programming modules. The system requires continuous monitoring to see if there are any deviations and if yes, then how they can be improved.

5.5.4 Post2-SSM Phase: Software Implementation

The SSDDD framework considers the domain model as the base from which the programming code is extracted by using the implementation pattern. Naked objects and TrueView are recommended as implementation patterns. A brief description of the implementation of PTS, as done by Joseph Ucizi Mtenje (2010), using both patterns is presented in the following sections. An evaluation of the implementation using both patterns, and a reflection on the framework as a development approach, are also provided.

5.5.4.1 Naked Objects implementation:

As discussed in Chapter 4, Naked Objects is an implementation pattern used as part of SSDDDF. The following is a sample of PTS implementation using the Naked Objects pattern.

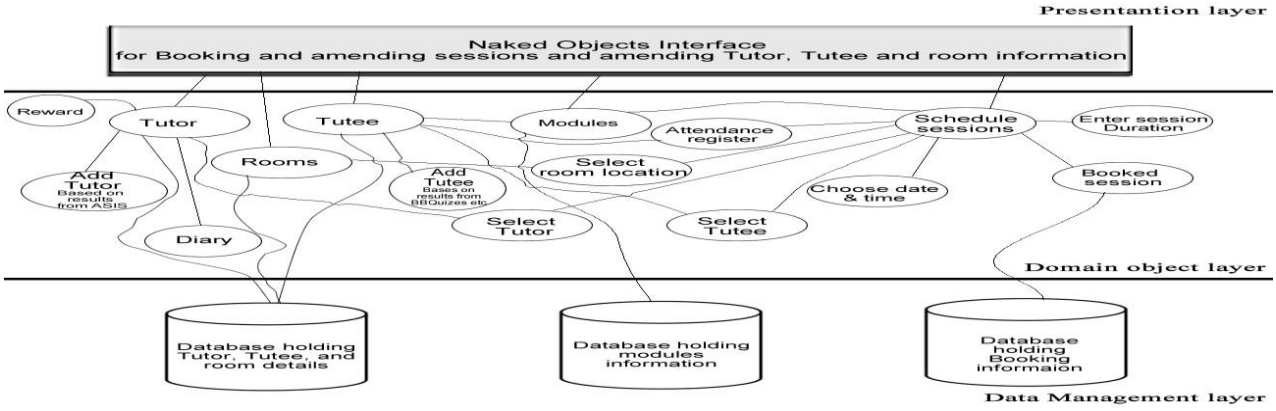


Figure5- 43: PTS Architectural Model Implemented with Naked Objects

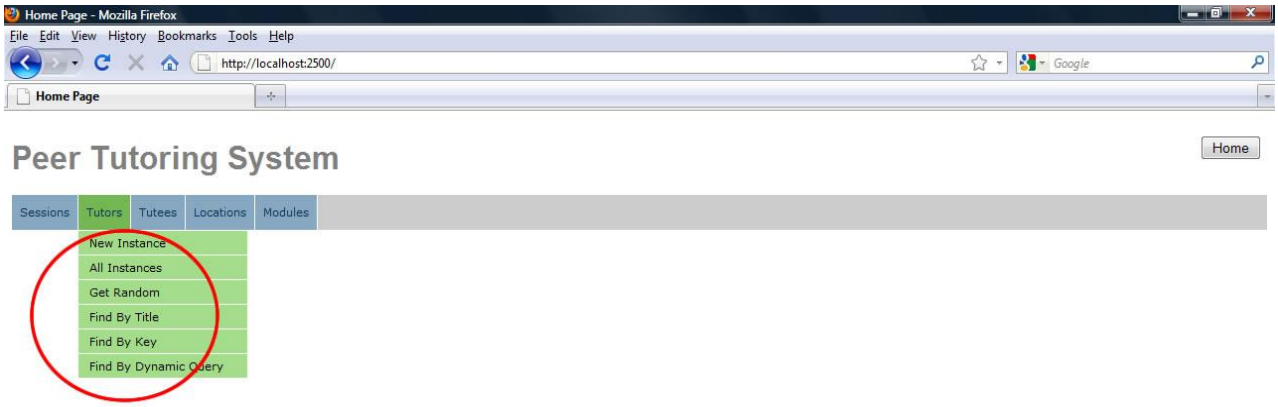


Figure5- 44: Naked Objects MVC Application

5.5.4.2 TrueView Implementation

The TrueView implementation pattern is suggested as an alternative to Naked Objects. The software is used to build an interface that users will use to access the system, to do all activities and arrange for sessions. The figures showing the user interfaces of PTS as implemented using the TrueView implementation pattern are provided in Appendix 8.

5.5.4.3 Evaluation of implementations

Joseph Ucizi Mtenje (2010) made a comparison between Naked Objects and TrueView as implementation patterns. The important issues he raised in this comparison was the usability of the system developed using Naked Objects and TrueView. He preferred Naked Objects over TrueView. The following section discusses the usability testing and the comparison between the two implementation patterns as presented by Joseph Ucizi Mtenje (2010).

1- Usability testing of TrueView prototype

When the TrueView application was created, a few users were requested to use it and provide the relevant feedback. The users were asked to perform different functions of the system like creating a tutor, tutee, new session, new location, and a module.

Some of them complained about the right click function, which is not commonly used by them in windows. Another user commented about the interface of the application that needs further improvement. On the other hand, one user liked the logic in the application that

allowed him direct access to the business objects and enabled him to manipulate them directly.

2- Usability testing of Naked Objects prototype:

Several users were also asked to try using the application developed with Naked Objects MVC, to comprehend its usability and user-friendliness. The users were asked to perform different functions of the system like creating a tutor, tutee, new session, new location, and a module. Also assign to module and mark attendance sheet.

One user commented that it was easy for him to use it but needed more improvement in terms of interface. Another lady user said that it was easy for here to manage it without training to perform all the functions. She added that it is easy for the users to navigate through the webpage. Other user also revealed that the system allowed them to search through the database by using different keywords.

3- Comparison between Naked Objects and TrueView patterns

Based on the above usability tests, the Naked Objects application was preferred by the users rather than preferring TrueView application (Ucizi Mtenje ,2010). The TrueView modeller does not support database integration, however, TrueView Agile Developer version supported it. For the PTS, supporting database integration is a necessity, so it would be essential to buy this agile version, which would mean greater cost to the client, while a better service can be provided more cheaply with Naked Objects MVC (Naked Objects, 2010). Nonetheless, as mentioned above, the usability of Naked Objects and its application interface that highly supports DDD are better than TrueView and therefore, it is more preferable than the TrueView.

5.5.4.4 Reflection on the SDDDF

In his evaluation, the postgraduate student Joseph Ucizi Mtenje (2010) mentions that he had not previously come across any combination like this. The closest one he had come across was that used by Lane and Galvin (1999), which combined and transited from SSM to object-oriented analysis, during which they moved from SSM conceptual models and developed use cases, but did not proceed to building an application using DDD implementation software. In SSDDDF, however, the application is built, allowing users to access business objects without using controllers, an aspect not mentioned by Lane and Galvin. Joseph Ucizi Mtenje (2010) adds that SSDDDF has many advantages, but the major one is that it enables the researcher to understand the problem situation better through

SSM, as it tends to provide different views of the situation from different stakeholders at the root definition stage, as well as at the DDD stage when it is important to understand the business objectives and how activities are done. This enables one to build a better application to suit the users' requirements, and also to build a system that more effectively fulfils the requirements that have been studied in the UML stage. The application will be easier to use, as it gives the user direct access to business objects and the facility to manipulate them more easily than through the controllers required in conventional MVC applications.

On the other hand, Joseph Ucizi Mtenje (2010) says that the point he found difficult in the framework was the point of conversion from SSM to UML, as this is not a one-to-one conversion, but involves the combination and decomposition of conceptual models. He advises that more research is needed in this area, in order to achieve a smoother and easier transition and to ensure that other researchers do not need to spend so much time it. This point will be considered in the discussion, and suggestions for future work will include the development of a pattern language to solve this situation.

5.6 Concluding Remarks

This chapter demonstrated the application of SSDDD framework in different case studies, which were taken from student's information system projects. After evaluating the application of the proposed system in both undergraduate and postgraduate projects, following concluding remarks/results are obtained:

1. The proposed framework is efficient in understanding the requirements off all the stakeholders at the initial stage, by comprehending the different perceptions and views. All the projects emphasized on this benefit of the framework. Through SSM, the system provides high clarity of requirements.

2. The proposed framework was efficient at understanding the soft and hard requirements of the information system, thus eliminating the major challenge that leads towards IS failure. It also provided high understanding of the problem situation through SSM.

3. The Naked Object implementation pattern, though time consuming and difficult to understand, provides high compatibility pertaining to DDD interface along with high usability. Also, the overall framework was found to be compatible with the other

implementation tools. However, a downfall of this approach is the lack of resources and time to understand it, as the timeframe for completing the projects by the students were insufficient.

4. With effective implementation of this framework, the project lifecycle can be improved, however, the developers need to be proficient to achieve this. Some of the students also professed that the utilization of this framework assisted them in improving their development skills.

5. The time was found to be a major constraint in adapting to the new framework. Therefore, it is advised to first provide training of the framework and then start with its implementation. Also, the resources unavailability was found to be a potential constraint

In conclusion, the requirements of the respective projects in case studies were efficiently identified through the deployment of SSDDD framework, thereby reducing the chances of IS failure. The system was also observed to be beneficial in terms of maintaining balance within the soft and hard requirements. The comparison of the current framework with the existing methodologies have been executed in the next chapter.

Chapter 6: Evaluating SSDDDF Through Teaching ISD module and the Comparison with other Frameworks

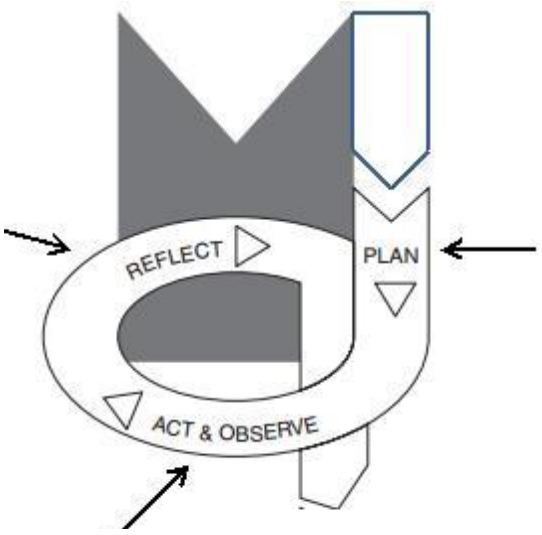
In chapter 5, the proposed SSDDDF was evaluated as an ISD development approach through practising different undergraduate and postgraduate students projects to gain the feedback and reflections from the students. This chapter is presented first the importance of the students feedback and reflections and why to use them to evaluate the proposed framework SSDDD through Action Research, and the justification of using the evaluation criteria and the evaluation framework. These are presented in section 6.1 and section 6.2 prospectively followed with section 6.3 which is presented further evaluation of the proposed framework by a larger sample of postgraduate students studying ISD module 'Methods and Modelling' in the Informatics Department in the University of Huddersfield. The comparison of SSDDDF with DDD and with other frameworks reviewed in the literature review chapter are presented in section 6.4 and section 6.5 prospectively. The comparison of the proposed and evaluated SSDDD framework with the existing studies was done to comprehend its contribution to the literature. In section 6.4, the comparison is made with the DDD framework, where the aim of the proposed SSDDD framework is to improve the DDD framework through modelling and implementing business domain systems, as well as by introducing a new language, named 'soft language', that enables the effective communication between different stakeholders of the system. This language is designed to operate as a complement to the 'ubiquitous language' of DDD. The framework has been evaluated through various case studies held at the educational setting, which has not previously been explored for DDD. Then section 6.5 presents a comparison of the proposed framework with the existing multi-methodology frameworks explored in the literature review chapter. These comparisons are briefly made, where the issues of the existing methods are depicted along with their solutions obtained through the current proposed methodology

6.1 The importance of Students Feedback and Reflections to Evaluate the planned Actions(The link between Action Research Evaluation approach)

6.1.1 Introduction

Soft systems approaches were categorized under action research approaches. In this thesis, action research has been adopted through the use of soft system methodology as a guiding methodology for the proposed framework. The use of different cases selected and explored within an educational background and using the framework for teaching ISD has allowed the current researcher, as a lecturer in the educational environment, to act as facilitator and action researcher during the research period. Coghlan and Brannick,2014 was mentioned that Action Research is a recursive process which allow the researcher to go through a cyclic process of planning, acting on the plan, reflecting on the outcomes, implementing the change and further re-planning. In the alignment of the literature of Action research, this thesis followed the cyclic process of evaluation in order to gain and refine the feedback and reflections of the students about the proposed framework. This process is re-planned and repeated different times using different case studies of ISD and by teaching and practising the framework tools through an integrated ISD case studies. Students feedback and reflections are important to support the formulation of the comparison criteria and the comparison process of SSDDDF with other methodologies and frameworks. By repeating the cyclic process of evaluation, the feedback and reflections were re-used by other cycles to improve the new feedback and reflections the next cycle, and the same were done for the followed cycles. Action Research as adopted methodology was illustrated in chapter 3 and presented following Kemmis & MC Taggart (2005) Action Research Spiral Fig(3-1) which is presented in chapter 3 and here.

Reflections on the benefits of the proposed framework and comparison with existing frameworks



Data gathering through literature review for identifying 'hard' and 'soft' criteria

Developing and evaluating the framework through interviews with stakeholders and developers (students) of ISD projects. Applying proposed technique to practical case studies. Teaching ISD using the proposed framework

6.1.2 The cyclic process of Action Research Execution

The cyclic process adopted by In this thesis performed and executed as follows:

1- Cycle1 : Plan: literature review to identify 'hard' and 'soft criteria. ******* Act & Observe: Develop the framework and practise it through illustrative case study**.***** Reflect: feedback and reflections from the students through the induction workshop.

2-Cycle2: : Plan: Prepare and submit 2 undergraduate case studies attached with the feedback and reflections from cycle1 to the 2 groups of students. *** Act & Observe: Apply the proposed framework to undergraduate practical case studies by the 2 groups.*** Reflect: feedback and reflections from the 2 undergraduate students groups through the practical case studies application.

3- **Cycle3:** Plan: Prepare and submit the first postgraduate case study to the first postgraduate student with feedback and reflections of first and second cycles. *** Act & Observe: Apply the proposed framework to the first postgraduate practical case study. *** Reflect: feedback and reflections from the first postgraduate student through the practical case study application.

4- **Cycle4:** Plan: Prepare and submit the second postgraduate case study to the postgraduate student with feedback and reflections of first, second, and third cycles.*** Act & Observe: Apply the proposed framework to the second postgraduate practical case study.*** Reflect: feedback and reflections from the second postgraduate student through the practical case study application.

5- **Cycle5**: Plan: Prepare the module 'Methods and Modelling' , the practical case studies and provide them to the students with the previous feedback and reflections.*** Act & observe: teach the module using the proposed framework and investigate the students through different data collection methods. *** Reflect: feedback and reflections from the postgraduate students done the module.

6- **Cycle6:** Plan: Formulate the comparison criteria based on the literature and the reflections about SSDDD gathered through all previous cycles to compare SSDDDF with DDD.*** present both frameworks performance into 2 separate tables to show their capabilities to handle different IS perspectives presented in the comparison criteria, develop

the comparison template, and compare SSDDD with DDD based on this template.*** Reflect: feedback and reflections from the researcher about the comparison results. 

7- **Cycle7:** Plan: Use the reflections about SSDDD gathered through the previous cycles to compare SSDDDF with other methodologies reviewed in the literature.*** present performance of SSDDD compared to each methodology.*** Reflect: feedback and reflections from the researcher. ⟶ Conclusion results and discussion.

6.1.3 Discussion and conclusion

The above cycles presented the evaluation of the action research utilized by this thesis. The feedback and reflections of students are very important since each cycle feedback will feed the next cycle to learn from the previous work and this represent the heart of the SSM as a guiding methodology of this reasearch. So, using the above link between the students evaluations through the practical case studies and teaching, the process will proceed to continue the action research evaluation by comparing SSDDD with DDD in section 6.4 and compring SSDDD with other methodologies in section 6.5 to recognize the capabilities OF SSDDDF among other frameworks documented in the literature. The action research was evaluated through the above formulated approach to gain the feedback and reflections of the students through the application of different case studies and teaching ISD module. The feed back and reflections are very important since these students acts as developers to evaluate the proposed framework. As a researcher and actor at the same time, I recognized that gathering feedback and re-use it for the next cyle is a good support to the next evaluatter in order to learn from the previous researcher efforts as learning is the heart of the adopted guiding methodology (SSM). By reaching the final cycle of students evlauation, their feedback and reflections became more clearer and benefecial to be used for the comparison with DDD and other methodologies. This approach wll guide this work to recognise the capabilities of SSDDD as an ISD approach and what is new about it.

6.2 Justifications of the evaluation framework

The evaluation framework adopted an evaluation criteria consist of different well known business perspectives (Table 6-1) where used to evaluate similar frameworks. The comparison done using this framework is limited to the availability of information about DDD and SSDDD, and the availability of judgment techniques. Likert scale is considered and used here to judge the contribution of each perspective of the proposed evaluation framework. Al Humaidan,2006 and others researchers used Likert scale before to judge

similar perspectives. May be other Judgment techniques will work better, but still the results obtained reflects good results about the evaluated framework SSDDD. The following subsections presents the justification of the selected criteria, the applicability of the criteria to gain results, the limitations of the adopted evaluation model, similar work used the same criteria, and the justification of the benefits of the proposed and evaluated framework SSDDD.

6.2.1 Justification of the selected criteria through the evaluation framework

The evaluation framework considered an evaluation criteria consist of different well known business perspectives where used to evaluate similar frameworks and added another two perspectives. Soft Perspective which used in similar comparison by Al Humaidan,2006 and widely applied in ISD by other researchers (Checkland,181; Avison,1990; Bustrad,1999, Petkov,2007;etc) and implementation perspective as a new one added by the proposed framework. The dependence of **soft perspective** is over the SSM techniques. These SSM techniques are responsible for the involvement of users in determining the roles of the stakeholder and the problem. The problems are verified using various means and before proceeding to the UML model, it is important to acknowledge the feedbacks and acceptance of the developed models. This involvement of SSM in DDD is not adopted as a consequence, being the availability of user involvement still the understanding of methods and techniques for the development of domain model was also not guaranteed. The handling of **organizational perspective** is through UML model technique and is done by both DDD and SSDDD. The benefit of SSDDD is that it uses both the use case and the class diagrams while only the class diagrams are used by DDD. **The behavioural perspective** is also handled more reliably using the SSDDD as it entails both SSM and UML model techniques as they indicate using the sequence diagram and activity diagram for modelling the activities depicted in use of case diagram. In this the descriptive modelling is performed using the UML diagrams whereas in DDD only class diagrams are used. As in the behaviour, it is not possible to fix it or standardize it as the directions can be changed on the basis of the occurrence of the various circumstances. **The informational perspective** is used to represent the informational entities required (entities within the structure and their relationships), and these can be presented in a tabulation form using use case proformas and class diagram. Through the evaluation, both DDD and SSDDD are not presented this perspective properly because some information is still not recognized by either of the

approaches, they cannot be considered complete. Based on this perspective the proposed approach supposed to develop or use other tools to represent informational perspective. **The functional perspective** is to handle the business process activities and information flow through SSM conceptual models and UML activity diagrams. Through this evaluation SSDDD used both techniques for modelling the business functions. This support using this perspective as part of this evaluation to be sure that the business process activities and information flow are modelled properly. The purpose of **the implementation perspective** is to handle the implementation of the domain model into an information system. Both approaches DDD and SSDDD done this using the implementation patterns to guide the developers and some of them considered this as a restriction of their choices. Based on this evaluation criteria, the main criterion of selecting the proposed framework was that it used both the SSM and the UML model techniques which give better outcomes.

6.2.2 Applicability of this Criteria gaining better results

The evaluation of the Information system development approach using specific evaluation criteria will help the evaluator to see the performance of the development approach and how it work. Here the DDD approach seeks the system process to be modelled as a domain model to be used for implementation. The basic concept of the DDD approach is the development of the ubiquitous language comprising of various types of concepts, designs, diagrams and documents in order to enhance and improve the domain experts and the developers' communications amongst them. These domain experts and the developers use this ubiquitous language for the purpose of developing and inventing a new domain model as stated by Evan in 2004. There are a number of diagrams which were used for modelling the business process as defined by UML. But the ability to solve and explore various issues related to problematic situations which can only be handled using the method known as soft system methodology (SSM) as described by Humaidan, 2006, Poulter, 2006 and Checkland, 1999. This SSM is a developed source of solving the problem which has its focus over the idealized model development of the systems which were relevant and comparable to the counterparts of the real world. The relationship amongst the SSM, design techniques and the object oriented analysis was defined by some researchers generally but, its applications are very limited. UML is considered to be the domain model by DDD. The developer is further guided by the SSDDD framework for the development of a Soft Language comprising of SSM output so that the soft aspects are dealt by them which are mishandled or not handled by DDD. The SSDDD works over the multimethodological framework handling both types of issues which are soft and hard of the business domain process modelling and

implementation as an IS. The level of understanding and clarity of stakeholders is very high as it applies to both type of soft and hard requirements successfully. The objectives are attained using the systematic approach decreasing the complexity and information system failures. The framework of SSDDD effectively manages and handles the changes. In the aforementioned ways the SSDDD attains better application results than the DDD approaches. This support the selected evaluation criteria to find out how all business (system) perspectives can be handled by the evaluated framework as the case of SSDDD.

6.2.3 Application of same criteria in similar work

The similar criteria is used in the other works as described by Al Humaidan,2006 and before by Curtiz,1992 and Warboys et al., 1999. They used the mentioned criteria to evaluate the workflow of the business process. Al Humaidan,2006 suggested the soft perspective to be added for evaluation the workflow system. The soft perspective is presented by the application of SSM techniques to develop the conceptual model which is mapped into UML diagrams. There are many extensions of the work which have been reported by several researchers. Like Penkov et.al, 2007 investigated the combination of SSM and UML extensions which comprising a systemic framework which was proposed by Penker et.al in 2000 for the purpose of modeling a business process of manufacturing factory. Wade et.al in 2009 described SSDDD as an approach for the development of information system seeking to model the system processes as domain model. The domain model developed by developers and the domain experts using the UL (Ubiquitous Language) of the DDD approach which supported the communications between several stakeholders. Various business process models were used a number of diagrams which are defined by UML and function as a part of SSDDD, but are unable to handle the soft issues related to the problematic situations. SSM usually can handle the problematic situation as stated by Evan, 2004 and Humaidan, 2006. The main purpose of using SSM in SSDDDF is to model business domain using rich pictures, conceptual model, and root definitions. Based on this clarification, the adopted evaluation framework combined the required criteria in order to handle all aspects related to the comparison between DDD and the proposed framework as an ISD approach..

6.3 Evaluating SSDDDF through teaching ISD module

This section presented the evaluation of the proposed framework SSDDDF through teaching ISD module 'Methods and Modelling' for Master students in the Informatics Department at the University of Huddersfield. The purpose of this evaluation part is to gain detailed

feedback and reflections about the framework tools after studying and practising them during the semester class work and assignments. This evaluation is to continue and repeat the cyclic process of Action research, as discussed in chapter three: plan, act and observe , and gain reflections. This evaluation part was used a group of methods of data gathering including In-Class surveys, reflective essays, analysis of common mistakes, and a feedback questionnaire. The aim is to collect more feedback and reflections from larger category of developers to support the framework comparison process with other frameworks.

Teaching business information systems modelling using UML will not lead to a complete understanding or enable the students or developers to implement a software system combining all the business experts' requirements. However, it may be argued that using an integrated framework in teaching business domain investigation and modelling can enhance understanding of such problematic situations and may lead to the development of a substantial software system. Based on this view, a group of MSc Advanced Computer Science and MSc Information Systems Management students, thirty eight, done the module 'Methods and Modelling' in September, 2011. The lecturer of the module was Dr. Steve Wade with the current researcher as teaching assistant. The module has been taught using the proposed framework SSDDD through practising it's different tools (SSM, UML, Naked Objects as an Implementation Pattern). By using this integration for teaching systems modelling, it was expected that the students would be able to see the whole systematic picture of the business domain, modelling would be understandable, and this would lead to a sufficient business domain model for coding the required software system. The evaluation techniques used during and at the end of teaching the module are presented in the following sections. This include In-class surveys, reflective essays, analysis of common mistakes, and feedback questionnaire.

6.3.1 In-class Surveys

Frequent in-class surveys were designed and used to evaluate the students' weekly satisfaction. This technique guided the teaching process in order to improve students' learning. This method depended on open-ended questions to obtain the students' feedback. From these it was apparent that the focus on identifying patterns to help students through difficult techniques was helpful. The majority of the students (approximately 60%) claimed to have had no prior experience of developing business models, but after completing the module, 86% said they felt confident with the use of soft systems techniques. There was

100% agreement that the ongoing feedback provided in this module was very useful. Typical comments included":

"I like the step-by-step approach where we move forward slowly with help at each stage. I think I would have become confused if I had to do all the work at the end."

"It helps to chunk up the work with patterns. Each pattern seems to make sense and when you put them all together you can make something happen."

As lecturers, we found that the approach taken was very time-consuming and might be difficult to implement when working with larger groups. Our focus on ways in which we could develop pattern-based teaching materials did lead us to spend more time looking at the students' work than we might otherwise have done. This helped us to see more clearly what techniques the students found hard to understand.

6.3.2 Reflective Essays

At the end of the course the students were asked to write a short reflective essay including a discussion about the module and how they used the techniques to develop their projects. This technique allowed the students to give their feedback about the techniques that they have been used. In addition, the evaluation had to include a wider discussion on topics such as:

How well the module related to other modules on their course? How the knowledge and skills taught on the module related to their previous experience as a student and/or employee? The appropriateness of the knowledge and skills taught on the module for future employment. Any particular aspects of the module that they found difficult. Specifically, any aspect of the real world that they wanted to capture in the models that they developed, any steps in the process that seemed to be a waste of time, or any additional steps that they thought might have been useful.

These essays provided generally positive feedback about the framework. The following comments are representative of some of the more general comments made in these essays:

"All of the techniques have proved very useful for me. I know how to design systems properly now."

"I have learned a lot from working in groups and following the method. I think this is the most important module because it links everything together."

"Before I started the module I did not know what modelling was or how it related to programming. I feel confident now that I can apply the techniques we have looked at on a real project."

Generalisations about the two groups were made and they were presented as follows:

- "The MSc Advanced Computer Science students were more comfortable with abstraction in the sequence and class diagrams. They seemed to regard modelling as high-level programming".

- "MSc Information Systems Management students were more comfortable seeing sequence diagrams and class diagrams as models of the real world".

In future presentations of the module it is proposed to create mixed groups so that each student gets to work with students on a different course.

6.3.3 Analysis of Common Mistakes in Classwork

The analysing of students final course work recognised different mistakes in their work. The purpose here was to find the reasons behind these mistakes and if they were related to the framework's techniques. This helped the researcher to determine how to improve the teaching of the module next time, and suggested an agenda for improving the SSDDD framework. A list of common errors would include the following:

- "Failure to use domain-specific terminology as presented in case study materials.
- Inconsistencies between sequence diagram and class diagram. For example, operations appearing in the sequence diagram that are not present in the class diagram.
- Operations given ambiguous names.
- Operations not supported by attributes or relationships.
- Database concepts (pk and fk) used in the domain model.
- A lack of consistency between the SSM models and the use case model".

As a future work, this work suggest the development of pattern languages that will steer future students away from making these types of mistake.

6.3.4 Feedback Questionnaire

A questionnaire is designed to further evaluate the proposed SSDDD framework as an integrated approach for information systems development. The design of the questionnaire is focused on the various components of the framework and the contribution of each to achieving the module's aim. The questions included in the feedback questionnaire were derived from the module's components and from the students' interaction during the course. Students' remarks and observations helped in the design of the questionnaire, which was used to evaluate the extent to which the module aim had been achieved.

The module's aim is: (To provide students with the knowledge and critical understanding of modern software and IS development methods, and skills to practice what they have learned in an integrated project). In teaching, there are different factors that may affect the achievement of any module aim. In the case of the 'Methods and Modelling' module for MSc students in the Department of Informatics at the University of Huddersfield, the investigation focused on one of these factors, which was the 'teaching approach' represented by the integrated SSDDD framework. It was believed that using the SSDDD framework, which combined different tools of systems modelling and development, would contribute to the achievement of the module aim. This framework was evaluated through teaching addition to the previous evaluation of it as an approach for information systems modelling and development. Since the aim of the module is clear, it was assumed that if the components of SSDDD framework were understood and practised effectively, then this would contribute to the achievement of the module aim.

At the end of the module, a feedback questionnaire was distributed among students to collect data about the contribution of each component of the framework to the achievement of the module aim. The Likert approach, which consists of five rankings, was used for this purpose: 5=Strongly Agree, 4=Agree, 3=Don't Know, 2=Don't Agree, and 1=Strongly Disagree. The data was analysed using SPSS statistical software. Means and standard deviation were proposed to analyse the descriptive data collected through 30 valid copies of the questionnaires out of 33 responses. The total number of students studying the 'Methods and Modelling' module between September 2011 and December 2011 was 38; 33 of them participated in this investigation, of which 30 responses were valid and used in this analysis. The results of the analysis are presented in the following section.

6.3.4.1 Feedback Questionnaire Data Analysis

To validate the contribution of understanding and practising of each component or activity of the framework in achieving the module aim, 'Means and Standard Deviation' were used for the different sections, each of which related to one component or activity. Tables 6-1, 6-2, 6-3, 6-4 and 6-5 present the descriptive analysis related to each component and activity respectively.

Table 6-1 shows that the means for the first component understanding and practising (SSM) were between 4.27 and 3.47. The highest mean was 4.27 for item numbers 1 and 3, which were "I found the tools of SSM were easy to use" and "I can see how SSM tools would help me to understand customer requirements", while the lowest mean was 3.47 for item number 8, which was "I am confident that I could use SSM conceptual models to depict the detailed logic of business processes". The arithmetic mean for all items related to SSM tools was 3.93.

Table 6-2 shows that the means for the second component understanding and practising (UML) were between 4.30 and 3.43. The highest mean was 4.30 for item number 1, which was "I found that UML is easy to use for modelling business processes", while the lowest mean was 3.43 for item number 4, which was "I found it easy to extract use cases from the SSM conceptual model". The arithmetic mean for all the items related to UML tools was 3.86.

Table 6-3 shows that the means for the understanding and practising the activity (linking SSM and UML) were between 3.83 and 3.57. The highest mean was 3.83 for item numbers 2 and 6, which were "I found that some of the activities in the conceptual model did not map directly to use cases" and "I found it useful to use SSM at the beginning to investigate the business domain and to move to UML and implementation", while the lowest mean was 3.50 for item number 4, which was "I found that the adopted method for transition is easy to use and practice". The arithmetic mean for all items related to linking SSM and UML tools was 3.67.

Table 6-4 shows that the means for the understanding and practising the fourth component(Implementation Pattern) were between 3.63 and 3.60. The highest mean was 3.63 for item numbers 1 and 4, which were "I found the implementation pattern is easy to adopt and use for implementation (Name of pattern :----------)" and "The interfaces generated by the implementation pattern are easy to use". The lowest mean was 3.60 for

item numbers 2 and 3, which were "I found moving from domain model (class diagram) to code is easy and not complicated" and "I found the implementation pattern easy to represent the domain model processes in code". The arithmetic mean for all items related to implementation pattern was 3.62.

Table 6-5 shows that the means for the fifth component(the integrated framework) were between 4.07 and 3.87. The highest mean was 4.07 for item number 4, which was "I found that this framework helped me to see an integrated picture of the required system in the project", while the lowest mean was 3.70 for item numbers 2 and 3, which were "I'm confident that this framework can be used to develop a complete software support system" and "I'm confident that all the systems components (soft and hard) can be investigated, modelled and implemented using this framework". The arithmetic mean for all items related to the integration of all components was 3.83.

Rank	No	Item	Mean	Standard Deviation
1	1	I found the tools of SSM were easy to use	4.27	.78
5	2	I can see how SSM tools would help me to understand the logic of business processes	4.03	.72
1	3	I can see how SSM tools would help me to understand customer requirements	4.27	.91
3	4	I can see how SSM tools could facilitate communication between business experts and developers	4.23	.90
8	5	I found it easy to understand and communicate with my team using SSM techniques	3.67	.80
4	6	I can see how an SSM Rich Picture can provide a comprehensive overview of a business system	4.20	1.00
6	7	I can see that SSM Root definition technique depicts the required system objectives	3.83	1.02
10	8	I am confident that I could use SSM Conceptual Models to depict the detailed logic of business processes.	3.47	.90
8	9	I can see how SSM conceptual models represent the business domain processes	3.67	1.03
7	10	I am confident that I could use SSM techniques to identify the user requirements	3.70	.79
		SSM Tools	3.93	.595

Table 6-1: Means and Standard Deviations Relating to understanding and practising SSM Component

Rank	No.	Item	Mean	Standard Deviation
1	1	I found that UML is easy to use for modelling business processes.	4.30	.88
3	2	I can see how Use Case diagram can be used to represent system processes.	3.97	1.10
7	3	I am confident that UML Use Cases are good tools for business process modelling	3.67	1.03
8	4	I found it easy to extract Use Cases from the SSM Conceptual model	3.43	.94
4	5	I found it easy to draw a sequence diagram based on each use case.	3.87	1.07
5	6	I found it easy to draw the Class Diagram based on the sequence diagrams.	3.87	1.04
2	7	I can see that UML Class Diagram represents the domain model of the investigated system.	4.10	.71
6	8	I understand how code can be generated from the domain model (Class diagram).	3.70	.99
		UML Tools	3.86	.618

Table 6-2: Means and Standard Deviations Relating to understanding and practising UML component

Rank	No.	Item	Mean	Standard Deviation
5	1	I found the transition from Conceptual Models to Use Case Models is an easy process	3.57	.94
1	2	I found that some of the activities in the Conceptual Model did not map directly to use cases.	3.83	.83
3	3	I can see that the resultant use cases represent the key activities of the conceptual model	3.70	.84
6	4	I found that the adapted method for transition is easy to use and practice	3.50	.57
4	5	I'm confident that I can depend on the resultant use cases to draw other diagrams like sequence and class diagrams	3.60	.89
2	6	I found it's useful to use SSM at the beginning to investigate the business domain and to move to UML and implementation	3.83	.91
		Linking	3.67	.517

Table 6-3: Means and Standard Deviations Relating to understanding and practising the linking of SSM and UML

Rank	No.	Item	Mean	Standard Deviation
1	1	I found the implementation pattern is an easy to adapt and use for implementation(Name of pattern:----------)	3.63	.89
3	2	I found moving from Domain model (class diagram) to code is easy and not complicated	3.60	.72
4	3	I found the implementation pattern easy to represent the domain model processes in code.	3.60	.62
2	4	The interfaces generated by the implementation pattern are easy to use.	3.63	.67
		Implementation	3.62	.429

Table 6-4: Means and Standard Deviations Relating to understanding and practising the implementation pattern

Rank	No.	Item	Mean	Standard Deviation
2	1	I found that integrating all the above tools in one development framework helped me to do the required project Easley	3.87	.78
3	2	I'm confident that this framework can be used to develop a complete software support system	3.70	.70
4	3	I'm confident that the whole systems components (soft and hard) can be investigated, modelled, and implemented using this framework.	3.70	.92
1	4	I found that this framework helped me to see an integrated picture of the required system in the project	4.07	.78
		Integrating	3.83	.631

Table 6-5: Means and Standard Deviations Relating to understanding and practising the framework as an integrated ISD framework

The above statistical analysis indicates that the SSDDD framework used to teach the 'Methods and Modelling' module can contribute to the achievement of the module aim as proposed at the beginning of this investigation. It indicates that the framework, as a guided learning approach, is acceptable as a framework for ISD. The results of this statistical analysis will be matched to the findings related to other techniques in the discussion (Chapter 7).

6.3.4.1 UML tools ranking

In relation to the second component of the framework(UML diagrams), question number 9 asked the students to identify which was the most important diagram among a given set: *"Which UML diagram do you believe is the most important one for business domain modelling among other UML diagrams?"* The students' answers to this were ranked from the highest to the lowest mean, and Table 6-7 shows the results.

Rank	No.	Item	Mean	Standard Deviation
1	1	Use Case Diagram	4.57	.73
2	2	Class Diagram	4.33	.84
4	3	Activity Diagram	3.70	.65
3	4	Sequence Diagram	3.80	.85
5	5	State Chart	1.97	.85
6	6	Collaboration Diagram	1.60	.93

Table 6-6: Most Important UML Diagrams from Highest to Lowest

Table 6-6 shows that the most important diagram for business domain modelling, among other UML diagrams, was considered to be the 'use case diagram', with a mean of 4.57 and standard deviation of .73, which is statistically significant. The lowest ranked diagram was the 'collaborative diagram', with a mean of 1.60 and standard deviation of .93, which is also statistically significant. These results are presented in Figure 6-1.

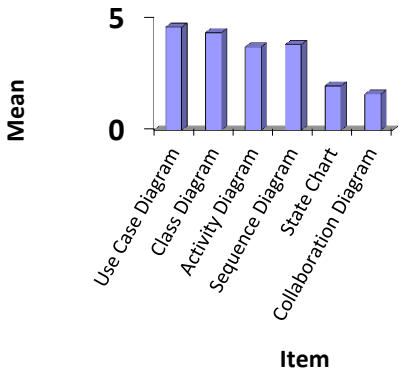


Figure 6- 1: Most Important UML Diagrams from Highest to Lowest

6.3.5 Conclusion

This part has been evaluated the SSDDD framework as an ISD approach through teaching the module 'Methods and Modelling' for Informatics Master students. ISD. This evaluation has adopted different tools to collect and analyse feedback data from the respondents. During the course, three tools were used to investigate the students; these comprised in-class surveys, reflective essays, and analysis of common mistakes in classwork. These techniques provided feedback from students that would be reflected in the comparison of SSDDD, future work, and any enhancement of the framework. In addition, the different types of mistakes and reasons behind them have been highlighted, and future work will try to address these issues. Finally, a feedback questionnaire was distributed to the students and analysed using SPSS software to focus on the importance of all tools of the framework separately and the framework as an integrated one. The statistical calculations focused on the contribution of the SSDDD framework to achievement of the 'Methods and Modelling' module's aim. The results indicates the acceptance of the framework as an ISD approach with different comments and remarks that will be for the future work. Detailed discussion and matching of all of these results will be presented in Chapter 7.

6.4 Comparing SSDDDF with DDD

In both the frameworks, business domain perspectives are modelled and implemented into an information system to support different organizational functions. It was discussed in Chapter 2 that business domains, and the information systems implementing them, consist of 'hard' and 'soft' perspectives. In order to make a comparison between DDD and SSDDDF, these perspectives have been formalized as described in the following section. This formalization enables these perspectives to be used as the basis of the first comparison, which considers the frameworks as approaches for modelling and implementing the business process perspectives of any business domain. The comparison will be presented as follows: section 6.3.1 presents the business domain perspectives(criteria), while sections 6.3.2 and 6.3.3 show how DDD and SSDDDF respectively handle each perspective through the modelling and implementation of a business domain. Finally, section 6.3.4 show how the proposed evaluation criteria is used to compare DDD and SSDDDF as an ISD approach with explanation how each perspective handled.

6.4.1 Business Domain Perspectives(Evaluation criteria)

As discussed in Chapter 2, various authors agree that the business process of any business domain comprises of different perspectives (Curtis, 1992; Warboys et al., 1999). These perspectives are discussed and summarised in Chapter 2, where they are identified as functional, organizational, behavioural and informational views. These have been adopted by other researchers and used to model and implement business processes of the business domain (Al Humaidan, 2006). This thesis will briefly present these perspectives and introduce a new 'soft perspective', as suggested by Al Humaidan (2006), to model the business process as a workflow system. In this research, the business process has been modelled using SSDDDF as a 'business domain system' to be used for implementation. Then, the way in which these perspectives are handled by both DDD and SSDDDF will be presented in tabular form. The comparison will use these tabulations to reach a conclusion about the performance of DDD and SSDDDF as approaches to modelling and implementing the business process of the business domain. The following table (6-7) represents business process perspectives 2-4, as presented by Curtis (1992) and Warboys et al. (1999), and adds the soft perspective (no. 1) proposed by Al Humaidan (2006) and Salahat et al. (2009), which includes SSM to model the soft perspective. In addition, the implementation perspective (no. 6) is proposed for including an implementation pattern. The soft and implementation perspectives included in this table are based on the notion of modelling and

implementing the 'business process of the business domain' as 'a business domain system'. In the below table (6-7), the perspectives 2-4 are by Curtis (1992) and Warboys et al. (1999).

6.4.2 Modelling and Implementing 'Business Domain' Perspectives using DDD

Chapter 2 explored the role of 'domain-driven design' as a software development approach to the investigation of modelling and implementation of any investigated business domain. It consists of different layers and aims that concentrates on the domain layer before the commencement of implementation. The different business process perspectives are presented in Table 6-7, where DDD can handle these perspectives up to different levels. All the business perspectives, except the implementation, belong to the domain layer. The other DDD layers (interface, application and infrastructure) belong to the implementation perspective. Thus, the domain layer contains the concepts of the business domain, business rules and use cases, state and behaviour of business entities and information about the business situation. The domain layer attempts to model the business domain into a 'domain model' that can be implemented through the implementation layer using any pattern. Table 6-8 presents the management of each business domain perspectives by DDD.

6.4.3 Modelling and Implementing 'Business Domain' Perspectives using SSDDDF

Systemic soft domain driven design framework (SSDDDF) is a new proposed framework designed to enhance the DDD approach by handling the soft issues of the business domain. This approach was demonstrated in chapter 4 and evaluated as a ISD approach in chapter 5 using different student projects. The results of these evaluations are used now and presented in a tabulation form. The application of the framework, and its capability of handling the processes within the business domain perspectives, is presented in Table 6-9. Based on this comparison of the two frameworks as development approaches, section 6.2.4 will show how the adopted evaluation framework is used to evaluate both approaches to understand the enhancement of SSDDD framework as compared to the existing DDD.

Perspective	Description
1- Soft	This perspective is added by this thesis to deal with soft aspects of the business process. For the SSDDD framework, this refers to the first two investigative phases: the pre-SSM phase to identify the problem and stakeholders' roles, and the SSM phase to evaluate the problem using SSM and produce 'soft language'. From this perspective, progression can be made to other perspectives through the transition process from SSM CPTM diagram to use cases. Different soft issues will be included, such as users' involvement in modelling and development of the system, different stakeholders' views, users' satisfaction, etc.
2- Organizational	This focuses on who will perform the business process activities and where (the organizational structure).
3- Behaviour	This perspective deals with the timing of the execution of business process activities (ordering), and how they can be executed.
4- Informational	Deals with the informational entities required (entities within the structure and their relationships).
5- Functional	Deals with business process activities and information flow.
6- Implementation	Deals with implementing the domain model into a software support system using a DDD implementation pattern.

Table 6-7: Business Process Perspectives

Perspective	How DDD handles each perspective
1- Soft	DDD partially handles soft issues through the usage of ubiquitous language as a means of communication between team members to avoid misunderstanding and an inconsistent model. However, UL does not include any soft modelling tools to allow the users to participate in the development or to provide feedback and agreement about the system activities being modelled. It only facilitates communications between team members.
2- Organizational	This relates to DDD's domain layer modelling of the business processes. This perspective focuses on who will perform the business process activities and where (the organization's structure). The DDD approach achieves this by developing the domain model, which is represented by a class diagram. However, there is no indication in the class diagram of who will carry out the activities presented in the domain model.
3- Behaviour	This also relates to DDD's domain layer modelling of business processes, but this perspective deals with the timing of the execution of business processes. The DDD approach achieves this by developing the domain model, which is represented by the class diagram.
4- Informational	Again this relates to DDD's domain layer modelling of business processes, but this perspective deals with the informational entities required (entities within the structure and their relationships). The DDD approach achieves this by developing the domain model, which is represented by the class diagram.
5- Functional	This relates to DDD's domain layer modelling of business processes, and this perspective deals with business process activities and information flow. The DDD approach achieves this by developing the domain model, which is represented by the class diagram.
6- Implementation	This relates to the implementation layer and deals with implementation of the domain model into a software support system using a DDD implementation pattern. There are different DDD implementation patterns available, such as Ruby, Naked Objects, TrueView, JMatter, XT Framework, etc.

Table 6-8: Handling of each Perspective by DDD

	Perspective	How SSDDDF handles each perspective
1-	Soft	Investigation starts with the pre-SSM phase to identify the problem and stakeholders' roles, followed by the SSM phase to evaluate the problem using SSM techniques and produce 'soft language'. These phases involve users by enabling them to express their views and participate in identifying the problem and the roles of stakeholders. They are then involved in analysing the problem by constructing the rich picture and the root definition. Next, users are involved in the development of different conceptual models to represent different stakeholders' views (human activities), and in the construction of the consensus primary task model (CPTM) which includes all the activities agreed by different stakeholders. Users can recognise how the system is presented in the CPTM and compare it to what they have used in the real life system. If any amendments are required or they are not happy with this model, the team will modify it until the users are satisfied. This involvement will promote acceptance of the software system that will be developed based on the SSM modelling techniques, as it can be understood more easily by users than other, more technical methods. From this perspective, progression can be made to other perspectives through the transition process from SSM CPTM diagram to use cases. Different soft issues are handled, such as users' involvement in modelling and developing the system, determination of different stakeholders' views, users' satisfaction, etc.
2-	Organizational	This perspective focuses on who will perform the business process activities and where (the structure), and the use case diagram represents these activities and their actors. In addition, this perspective can be modelled using the class diagram by assigning tasks to users using the role concept.
3-	Behaviour	Since this perspective deals with the timing of the execution of business processes, the sequence diagram (timing) and activity diagram are used to model all activities depicted in the use case diagram. The SSM conceptual model deals with this perspective partially, but detailed modelling is done by UML (sequence and activity) diagrams.
4-	Informational	This perspective deals with the informational entities required (entities within the structure and their relationships). The tabulation of activities, presented in use case proformas, and the class diagram are used to model this perspective
5-	Functional	Since this perspective deals with business process activities and information flow, these activities are depicted in SSM conceptual models and modelled using the UML activity diagram.
6-	Implementation	This deals with implementation of the domain model into a software support system using a DDD implementation pattern. SSDDD recommends Naked Objects or TrueView as implementation patterns.

Table 6-9: Handling of each Perspective by SSDDDF

6.4.4 The application and using the evaluation framework to Compare DDD with SSDDDF as an 'Information Systems Development' Approach

DDD and SSDDDF were compared on the basis of the modelling and implementation of 'business domain' perspectives. It was discussed in Chapter 2 that business domains, and the information systems implementing them, consist of 'hard' and 'soft' perspectives. In order to make a comparison between DDD and SSDDDF, these perspectives have been formalized as described in Table 6-7 which presents a summary of these perspectives. DDD was discussed and described in chapter2, and these information are used now to see how DDD handle the business perspectives(the comparison criteria) which is presented as a tabulation form in Table 6-8. The proposed framework SSDDD is evaluated and demonstrated with a case study in chapter 4 and further evaluated as an ISD approach through different students projects in chapter 5 and valuable information were obtained through this evaluation. In this chapter, the framework is re-evaluated through teaching ISD module and valuable information were gained and used during the comparison process. First, these information were used to see how the SSDDD framework handled the proposed evaluation criteria (business perspectives) and it is presented as a tabulation form in Table 6-9. By Using the information obtained and presented in Table 6-8 and Table 6-9, the comparison between the DDD and SSDDD was presented in Table 6-10 based on the utilized comparison schema. The schema used to compare DDD and SSDDDF was developed based on the research of Al Humaidan (2006) and Likert scale values. The current research utilizes this means of comparison, as it provides a clear and precise information required to assess the performance of the proposed mechanism. The schema considered Likert scale values to be assigned both to DDD and SDDD based on their ability to handle the related issues of any given perspective(soft perspective, organizational perspective,…etc).The schema was defined as follows:

1- 4 points: if the framework handles all issues of the business domain perspective

2- 3 points: if the framework handles more than half of the issues of the business domain perspective

3- 2 points: if the framework handles at least half of the issues of the business domain perspective

4- 1 point: if the framework handles less than half of the issues of the business domain perspective

5- 0 points: if the framework does not handle any of the issues of the business domain perspective

Perspective	Business Domain Modelling and Implementation Approach/Framework	
	DDD	SSDDDF
Soft	3	4
Organizational	4	4
Behaviour	2	3
Informational	3	3
Functional	3	4
Implementation	3	3
Total	18/24	21/24

Table 6-10: Comparison between DDD and SSDDD

Firstly, neither approach can be considered as 100% perfect for the information system development. Further improvements can be made via rigorous investigation of the issues. The allocation of points and different perspectives are explained and justified below:

1- The soft perspective is entirely dependent on SSM techniques, which support the users' involvement in determining the problem and stakeholders' roles, and investigating the problem through the development of rich picture, root definition, conceptual models and the CPTM. The use of feedback and acceptance of the models being developed is important before proceeding to UML modelling and DDD implementation patterns. Based on this, SSDDD was given a score of 4. In contrast, DDD does not adopt SSM. Thus, while user involvement is still available, it cannot be guaranteed that the users will be able to understand all the methods and techniques used to develop the domain model. It is estimated that users may be able to understand half of these but not all, so the score given here is 3.

2- The organizational perspective is handled by both DDD and SSDDD through UML modelling techniques. Since this perspective focuses on who will perform the business process activities and where (the organizational structure), the use case diagram represents these activities and their actors. In addition, this perspective can be modelled using the class diagram by assigning tasks to users using the role

concept. SSDDD utilizes use case and class diagrams, while DDD uses only class diagrams. Both approaches are therefore given 4 points because they model this perspective using UML tools.

3- The behavioural perspective is handled by SSDDD through SSM and UML modelling techniques. Since this perspective deals with the timing of execution of business processes, the sequence diagram (timing) and activity diagram are used to model all activities depicted in the use case diagram. The SSM conceptual model deals with this perspective partially, but detailed modelling is done by UML (sequence and activity) diagrams. In contrast, DDD depends only on the class diagrams, which can show the behaviour of these activities but is more reliant on data, such as entities, types of data, data structure, etc. For this reason, SSDDD is given 3 and DDD is given 2. This research believes that the behaviour cannot be standardized or fixed, as a variety of circumstances may occur which cause the change of direction.

4- The informational perspective deals with the informational entities required (entities within the structure and their relationships), so the tabulation of activities presented in use case proformas and class diagram are used to model this perspective. Both DDD and SSDDD use the UML class diagram to model this perspective. Based on this, 3 points are given for both the approaches. As some information is still not recognized by either of the approaches, they cannot be considered complete.

5- The functional perspective deals with business process activities and information flow, and these activities are depicted in SSM conceptual models and modelled using the UML activity diagrams. The SSDDD framework models this perspective using both SSM conceptual models and the UML activity diagram, but DDD depends on the class diagram, which partially or indirectly depicts these functions. Because of this, SSDDD is given 4 points while DDD is given 3 points.

6- The implementation perspective deals with implementation of the domain model into an information system using a DDD implementation pattern. SSDDDD considers two DDD implementation patterns, Naked Objects and TrueView, while DDD leaves it open for users to select the implementation pattern from a range of different available patterns. Based on this, both SSDDD and DDD perform the implementation perspective and because of this, both are given 3 points. However, some of the students who developed projects during the evaluation period complained about

SSDDD restricting them to the use of these two implementation patterns; they said the choice of options should be kept open because it would take them more time to master new patterns.

Overall, SSDDD earned 21 out of 24 points while DDD earned 18 out of 24 points. Therefore, the enhancement of DDD as an information system development approach was achieved. The improvement percentage was calculated as follows:

The performance of SSDDD was calculated as 21*100/24=87.5%, while that of DDD was calculated as 18*100/24=75%. Thus, the percentage of improvement to DDD by adopting the new SSDDD framework as an information system development approach is 87.5%-75% = 12.5%. There are various areas in which further improvement can be achieved, and these are presented in Chapter 7 in the form of recommendations and suggestions for future work.

6.5 Comparing SSDDD with Existing ISD Approaches

The proposed framework SSDDD is mainly compared to DDD and a criteria is applied since the purpose of this work is to see if the SSDDD enhanced DDD. Also, brief comparisons of SSDDD and other ISD models discussed in chapter2 were done here to see how SSDDD is different and to link it to the existing knowledge.

6.5.1 Comparison with SSADM (Structured Systems Analysis and Design Method)

SSADM, a traditional methodology, is well-structured but has several drawbacks. The method places considerable emphasis on planning and analysis, which requires eminent time and cost before constructing an information system. From a management perspective, the approach allows rigorous planning and prediction of schedule and budget for the system development. However, it may be argued that because this approach requires the project manager to plan a lot of the work and activities involved in the system's development, this will take a lot of time and then there may be problems in making any changes to what has been planned. It also places less emphasis of the changing requirements and has less flexibility in the framework. Moreover, the understanding of the framework is difficult and requires initial training and learning for effective utilization.

On the other hand, the proposed SSDDD framework places adequate amount on planning while focusing more on requirement analysis, thus creating room for any modifications in the future, as per the requirement changes. Also, the framework is easier to understand,

though requiring learning, it may be comprehended with less difficulties, as inferred from the current case studies.

6.5.2 Comparison with Agile Methodologies

A number of development methods have been proposed, which use UML with varying degrees of agility. One of agile methodologies is 'Extreme Programming-XP' which emphasizes on iterative and incremental development methods and provides explicit and hands-on methods for developers. Another agile methodology is 'Feature-Driven Development-FDD' which is developed by Jeff De Luca (1997). FDD is a management-supporting tool that suggests a specific framing of the process as well as iterative development, but does not provide guidance in respect to specific development methods.

However in the existing agile methodologies, all the modern development methods recognize that business software requirements are highly volatile. This approach is flawed because users increasingly find themselves in changing business situations and are therefore unable to identify unalterable requirements. The model of software development as an adaptive process, in which detailed requirements emerge iteratively as a project progresses and are modified as learning takes place, seems much more appropriate. These methodologies focus on making the development process shorter than traditional hard approaches. However, none of these, nor any of the others, have tried to solve the problem of soft system aspects.

Therefore there is a need for a methodology that has increased emphasis on 'use cases' and 'iterative' development techniques. Use case is referred to as a piece of functionality that provides meaningful value to a user. The current methodology (SSDDDF) integrates UML with SSM and utilizes use cases to deal with the dynamic user requirements in the most efficient manner.

6.5.3 Comparison with Multiview methodology

Both the soft and hard aspects of building the system are incorporated in the Multiview methodology by working in alignment with the soft system methodology and Yourdon Systems Modeling. The major constraint of Multiview methodology is that it is unable to provide the tools and techniques to be used for implementation of the information system. Also, it provides less flexibility between the different phases with inconsiderate thought on how to iterate between the stages. The proposed SSDDD framework is efficient in these terms and provides higher flexibility. It offers implementation tools that are compatible with

the other components of the systems. However, this can also be a drawback as the proposed methodology offers only two options (Naked Objects and Trueview) of implementation patterns.

6.5.4 Comparison with SWM (Soft Workflow Methodology)

The soft workflow methodology addresses only two major concepts, which are organizational business processes and workflow system modelling, the rest of the process is structured and if managed inefficiently, can lead to system failure. Also, the approach is not evaluated or verified using case studies, therefore, having no real time application to judge its performance. Without the implementation of the framework to a single case study, the SWM method cannot be generalized to other situations. Apart from these, the framework fails to incorporate all the eleven perspectives of the workflow system (as mentioned in the research by Al-Humaidan, 2006). The framework handles few of these perspectives along with soft perspective. The reason behind this is that the framework has not been applied in the real world scenario.

The proposed methodology surpasses this issue as it addresses all the mentioned perspectives by implementing the framework in the real case studies. All the case studies, peer tutoring system, school's liaison coordination system and student association system have evaluated all the perspectives while emphasizing on the user requirements of the information system.

6.6 Conclusion

This chapter has presented the importance of the students feedback and reflections to evaluate the planned actions through action research, and the justification of the select criteria and framework to compare the SSDDDF with DDD and the existing frameworks. Then this followed with further evaluation through teaching the module as an ISD for Master students and feedback and reflections were collected through different data gathering techniques. Then, the SSDDDF is compared with DDD as an information systems development approach. The comparisons between the proposed methodology and the existing multimethodologies have been presented to comprehend the contribution made by the current study. This comparison is a part of the process of evaluating SSDDDF which has been considered in Chapters 4 and 5, and now in Chapter6. The results of the SSDDDF evaluations presented in all these chapters (4, 5 and6) will be combined and discussed further in Chapter 7.

Chapter 7: Conclusion

7.1 Introduction

The present research has investigated improvements to domain-driven design (DDD), as an information system development approach, by considering both, soft and hard perspectives of the business domain. As a result of this investigation, a new framework has been proposed and evaluated as an approach for ISD development. The framework is named 'Systemic Soft Domain-Driven Design', and it combines soft system methodology, unified modelling language and a domain-driven design implementation pattern to address business domain perspectives. This chapter provides an overview of all the results of SSDDDF evaluation, followed by a discussion of these results. Then, the contribution of this research is conceptualized and explained. Finally, the limitations of the new SSDDD framework and recommendations for future work are presented, followed by the concluding remarks.

This thesis has proposed and developed the SSDDD framework as an approach to information system development. The research aimed to answer the two research questions in order to fill the aforementioned gaps in knowledge. These research questions are:

Q1: How can we formulate a multimethodology framework that will allow us to investigate, analyse, model, and implement the business processes from a specific domain by considering all the relevant "soft" as well as "hard" system requirements?

Q2: What benefits can we demonstrate from applying the proposed framework in a number of ISD projects?

The tow gaps in knowledge, as determined and summarised in Chapter 2, are as follows.

Gap 1: this research builds on the framework presented in 'Domain-Driven Design' (Evans, 2004) but, as the author has disclosed, there is room for improvement in the 'ubiquitous language'. With DDD, the stakeholders participating in project development may not understand the methods and techniques used due to language constraints, and this is related to their education and work-based experience. This raises the question of whether it is possible to eliminate these difficulties through the adoption of the proposed development framework – SSDDD by developing a soft language..

Gap 2: one methodology or framework may not be enough to develop a system. All information systems development methodologies have limitations, and it is expected that these methodologies can be improved in the future (Avison et al., 1990). This thesis has tried to improve DDD by understanding and inculcating both the soft and hard requirements in ISD.

Different stages of evaluating SSDDD have been undertaken over the course of several years. The framework has been evaluated and compared with DDD and other ISD methodologies and frameworks as an approach to ISD. The evaluation work is discussed and presented in Chapters 4, 5 and 6 and is now combined and overviewed in this chapter, followed by a discussion and consideration of the contribution of this research.

7.2 Results and Discussion

7.2.1 Evaluating SSDDD as an ISD Development Framework Through Different ISD Projects

Following the literature review, the researcher of this thesis has proposed and explained the SSDDD framework, and illustrated it through the PTS case study. The illustration shows how the framework can be used and applied for developing information systems. Then, the framework has been evaluated again as an ISD framework through different real life projects undertaken by undergraduate and postgraduate students. Two undergraduate projects, and another two postgraduate projects, have been presented as a means of evaluating the framework as an ISD approach, and feedback from the developers about the application of the framework is given in Chapter 5. This feedback, together with evaluative comments, is presented in the following subsections as a summary of these evaluations.

7.2.1.1 PTS Development - Feedback from Undergraduate Students

As mentioned in Chapter 5, a group of students selected the PTS system as a graduation project to be developed using the SSDDD framework. After the students completed their project, they were asked to provide feedback about the application of SSDDD framework. Following benefits are revealed:

- Clearer definition of requirements through investigation using soft system methodology (SSM);

- High commitment to the object-oriented approach using UML and the Naked Objects framework;

- Shorter project lifecycle as requirements are clearly identified from the beginning, thanks to SSM.

7.2.1.2 Discussion on SSDDD as a framework for PTS undergraduate project

This reflection, based on the students' achievements, supports the argument for using the proposed framework as an information system development approach, as it enables the understanding of both soft and hard issues of the system being investigated. The students stated that the system requirements were clearer for them because of using SSM at the beginning, which makes the time required for development shorter. In addition, they supported the usage of UML as a modelling approach to model the business domain, which can then be implemented using the Naked Objects implementation pattern. In alignment with this result, as per the literature review, according to Lucky & Adegoke (2014), the challenges faced in the development of information systems correspond to the infrastructures (both hardware and software), and lack of understanding of the user requirements. The researchers have further determined that developing a complex information system requires a multimethological approach that is rendered as the most effective strategy. According to Al-Humaidan (2006), both SSM and UML must be used to address the hard and soft components of a system and thus increases the clarity of requirements.

As professed by Xia & Lee (2005), the dynamic business requirements and organizational needs have created difficulties in developing a system that fulfils all the requirements and system specifications. Therefore, an information system must be developed that is able to comprehend all the requirements of stakeholders and organizational goals. The understanding of soft aspects and integrating it with technical aspects ensures the success of a project as it addresses the specific needs required from the system. According to Kaur & Aggarwal (2013), high competitive environment has compelled the organizations in improving their information systems for meeting the demands of the emerging markets, as a lack of understanding leads to ISD failure. Understanding the business needs and inculcating them in the development of information systems contributes to the successful compilation of the system without any failure. Integrating hard and soft approaches ensures the same (Hasan, 2003). In the current research, both the hard and soft approaches have

been integrated to develop the system, and the results demonstrate the success of the application, thus supporting the literature.

7.2.1.3 Students Association System (SAS) - Feedback from Undergraduate Students

The students reported that applying the SSDDD framework helped them to improve their development and documentation skills. However, they raised the issue that the time framework allowed to complete this project was not suffient, since they needed to explore different aspects of Naked Objects, as it was new to them, and required more practice to improve their professional development. They agreed that applying the framework as an integrated approach for information system development was good, but that the required resources must be available, especially original copies of Naked Objects rather than trial versions. They also said that the software they had developed was a prototype and would need further enhancement and refinements in the future. They hoped to improve the system so that it could be available online for any member to access remotel.

7.2.1.4 Discussion on SSDDD as a framework for SAS undergraduate project

It is not easy for all the students at junior developer level to deal with Naked Objects, but if given enough time, some of them will handle it well. However, the students agreed that their development and documentation skills were improved by applying the SSDDD framework. They also supported the idea of using an integrated framework for developing. They focused on the resources required to use the framework, which must be available and mastered in advance in order to develop the system properly. According to Avison and Wood-Harper (1990), it is essential to provide tools and techniques in a framework to promote efficient implementation of the information system. The ISDM that are unable to handle the information systems perspectives (both 'soft': "human-centred" and 'hard': "technology-centred") causes the IS failure (Barjis, 2008).

The current analysis is in alignment with the literature as it offers implementation tools, however, using them needs further learning by the developer. Also, the SSDDD framework deals with the hard and soft requirements that helps in facilitating the development skills of the students, as they are able to work with different perspectives. Warboys, Kawalek, Robertson, and Greenwood (1999) stated that the business process can be defined from different viewpoints, which are the functional view, organizational view, behavioural view and informational view. In the current research, all these views are addressed that assists

the success of the framework and reduces the chances of failure. Therefore, the existing studies have supported the result obtained in the present investigation.

7.2.1.5 Schools Liaison Coordination System (SLCS) - Feedback from Postgraduate Student

The postgraduate student Saraj Din (2009) explains that the purpose of adopting SSDDDF was to discover if he could use it to develop a software application. In his evaluation, Saraj Din (2009) mentions that SSDDDF enables the researcher to understand and explore the problem situation better through SSM. It enables him/her to gain different views of the current situation through the stakeholder analysis and root definition modelling stages. This can facilitate an understanding of the business objectives and how activities are done. It enables the developer to build a better application that suits the users' requirements, and even to build a system that improves on those requirements. The UML stage helps the user to model the system well and to understand the system requirements exactly. However, he adds that it was difficult for him to use Naked Objects because of the unavailability of resources, and he himself was not prepared to implement the software using the Naked Objects implementation pattern because of the time required to master it and to obtain the resources.

7.2.1.6 Discussion on SSDDD as a framework for SLCS postgraduate project

Looking at the above mentioned problems, it is evident that they are not related to the nature of the framework, but to the developer himself. Such problems can be solved before starting any project by ensuring that developers are ready to use the framework completely, not partially as happened with Saraj. On the other hand, this point can also be regarded as a positive outcome, because it means the framework is compatible with the use of other tools for implementation, as happened in this case. This indicates that the framework can be applied to ISD projects and then other implementation approaches may be used, rather than the recommended patterns. Also, the current research revealed that SSDDD framework provides a better understanding of the problem situation due to the incorporation of SSM. This is in alignment with the literature, where Checkland & Scholes, (1990) have stated that SSM is a problem-solving methodology which focuses on the soft issues of a system and is applied to investigate problematic situations. Checkland & Howell (1998) have also observed the same aspect, that the use of SSM gives high clarity of the problem and issues in the system that reduces the chances of information system failure.

Therefore, it is inferred that the current research is supported by the literature, where the integration of SSM solves the emerging problems of ISD. Also, the soft language developed in the research is useful in providing more clarity and thus, compatibility with the system.

7.2.1.7 PTS - Feedback from Postgraduate Student

In his evaluation, the postgraduate student Joseph Ucizi Mtenje (2010) mentions that he had not previously come across any combination like this. The closest one he had come across was that used by Lane and Galvin (1999), which combined and transited from SSM to object-oriented analysis, during which they moved from SSM conceptual models and developed use cases, but did not proceed to building an application using DDD implementation software. In SSDDDF, however, the application is built, allowing users to access business objects without using controllers, an aspect not mentioned by Lane and Galvin.

Joseph Ucizi Mtenje (2010) adds that SSDDDF has many advantages, but the main one is that it enables the researcher to understand the problem situation better through SSM, as it tends to provide different views of the situation from different stakeholders at the root definition stage, as well as at the DDD stage when it is important to understand the business objectives and how activities are done. This enables one to build a better application to suit the users' requirements, and also to build a system that more effectively fulfils the requirements that have been studied in the UML stage. The application will even be easier to use, as it gives the user direct access to business objects and the facility to manipulate them more easily than through the controllers required in conventional MVC applications.

On the other hand, Joseph Ucizi Mtenje (2010) says that the point he found difficult in the framework was the point of conversion from SSM to UML, as this is not a one-to-one conversion, but involves combination and decomposition of conceptual models. He advises that more research is needed in this area, in order to achieve a smoother and easier transition and to ensure that other developers do not need to spend so much time on it. This point will be considered in the discussion, and suggestions for future work will include the development of a pattern language to address this situation.

7.2.1.8 Discussion on SSDDD as a framework for PTS postgraduate project

This student did a good job, especially in terms of exploring the transition process from SSM to UML through the conversion from CPTM to use cases. As he said, he found that this approach was not easy and needed more time. Regarding this point, this thesis believes that the solution to this problem is through the development of a pattern language which can be used to overcome the difficulty. This will be discussed further in the 'Future Work' section. Other feedback related to development and implementation encouraged the usage of the SSDDD framework as an ISD approach. The revelations of this case study is similar as before, where the proposed framework provided high clarity towards problem situation.

As per the literature review, according to Al Humaidan (2006), SSM is an approach to business process modelling that can be used for both general problem solving and management of change. The approach has been most successful in the analysis of complex situations where there are divergent views about the definition of the problem (i.e. 'soft problems'). Therefore, this approach assisted the student in developing the information system. Considering the difficulty in transition from SSM to UML, Galvin and Lane (1999) have mentioned that transiting from SSM to UML use cases imposes a problem as these methodologies are based on different paradigms ('soft' and 'hard'), and will be difficult for mapping the information gathered by the first methodology to the other one. In alignment with this study, the current research found that the postgraduate student identified this problem and required more time to make appropriate transition from one methodology to another.

7.2.2 Evaluating SSDDD as an ISD Development Framework Through Teaching ISD module

7.2.2.1 'Methods and Modelling' Module Teaching - Feedback from postgraduate students

The SSDDD framework has been re-evaluated as an ISD approach through teaching information systems development module 'Methods and Modelling' for a group of postgraduate students using the proposed framework SSDDD. The purpose here is to verify the previous evaluation results ,gained from chapter 5, through collecting and analysing more feedback and reflections from larger category of developers (postgraduate students). Each student was asked to select one project a mong a group of projects to practise the framework tools. The feedback and reflections were gathered from the postgraduate students through different investigation techniques including In-Class Surveys, reflective

essays, course work analysis, and feedback questionnaire. In-class surveys were used to evaluate student satisfaction on a week-by-week basis. The majority of the students (approximately 60%) claimed to have had no prior experience of developing business models, but after completing the module, 86% said they felt confident with the use of soft systems techniques. There was 100% agreement that the ongoing feedback provided in this module was very useful". From these it was apparent that the focus on identifying patterns to help students through difficult techniques was helpful. A reflective essay for the final part of the coursework portfolio, students were asked to write a reflective essay including a discussion on how the module reinforced (or otherwise) their appreciation of the techniques and processes employed in undertaking a development project. These essays provided generally positive feedback about the framework. The following comments are representative of some of the more general comments made in these essays: "All of the techniques have proved very useful for me, and I know how to design systems properly now" ."I have learned a lot from working in groups and following the method, and I think this is the most important module because it links everything together.". "Before I started the module I did not know what modelling was or how it related to programming, but I feel confident now that I can apply the techniques we have looked at on a real project" . Based on this feedback and reflections, certain generalisations about the two groups done the module can be made as follows:

- The MSc Advanced Computer Science students were more comfortable with abstraction in the sequence and class diagrams. They seemed to regard modelling as high-level programming.
- MSc Information Systems Management students were more comfortable seeing sequence diagrams and class diagrams as models of the real world.

In future presentations of the module it is proposed to create mixed groups so that each student gets to work with students on a different course. Analysis of the coursework submitted by the students revealed a number of common mistakes. A list of common errors would include the following: (Failure to use domain-specific terminology as presented in case study materials, inconsistencies between sequence diagram and class diagram; for example, operations appearing in the sequence diagram that are not present in the class diagram, operations given ambiguous names, operations not supported by attributes or relationships, database concepts (pk and fk) used in the domain model, and a lack of consistency between the SSM models and the use case model". A feedback questionnaire was distributed to the students and analysed using SPSS software to evaluate the

importance of each part of the framework and the framework as an integrated approach for ISD. The statistical analysis focused on the contribution of the SSDDD framework to the chievement of the 'Methods and Modelling' module's aim. Statistical analysis through the 'Main and Standard Deviation' calculations presented the importance of each components of the Framework and the integration of all of them in one ISD approach which supported the previous evaluations finding.

7.2.2.2 Discussion on 'Methods and Modelling' Module Teaching

While the module was running, the use of in-class surveys on a weekly basis helped the researcher to know that the majority of students were confident about using soft systems methodology to model the business domain. This supports the argument of this research that combining SSM with other methods will support systems development and facilitate a better understanding of the business domain. With regard to the final reflective essays prepared by the students, the majority of them stated that the techniques embodied in the framework were very useful for them, supporting them as they learned to work within groups and became ready to undertake a complete project. Finally, looking at the final work produced by the students, and considering the different mistakes they had made, supported recommendations for improvements to the module in the future. These feedback and reflections, based on the students' achievements through the period of the module teaching, supports the argument for using the proposed framework as an information system development approach, as it enables the understanding of both soft and hard issues of the system being investigated. The students stated that the system requirements were clearer for them because of using SSM at the beginning, which makes the time required for development shorter. In addition, they supported the usage of UML as a modelling approach to model the business domain, which can then be implemented using the Naked Objects implementation pattern. In alignment with this result, as per the literature review, according to Lucky & Adegoke (2014), the challenges faced in the development of information systems correspond to the infrastructures (both hardware and software), and lack of understanding of the user requirements. The researchers have further determined that developing a complex information system requires a multimethological approach that is rendered as the most effective strategy. According to Al-Humaidan (2006), both SSM and UML must be used to address the hard and soft components of a system and thus increases the clarity of requirements. As professed by Xia & Lee (2005), the dynamic business requirements and organizational needs have created difficulties in developing a system that fulfils all the requirements and system specifications. Therefore, an information system

must be developed that is able to comprehend all the requirements of stakeholders and organizational goals. The understanding of soft aspects and integrating it with technical aspects ensures the success of a project as it addresses the specific needs required from the system. According to Kaur & Aggarwal (2013), high competitive environment has compelled the organizations in improving their information systems for meeting the demands of the emerging markets, as a lack of understanding leads to ISD failure. Understanding the business needs and inculcating them in the development of information systems contributes to the successful compilation of the system without any failure. Integrating hard and soft approaches ensures the same (Hasan, 2003). In the current research, both the hard and soft approaches have been integrated to develop the system, and the results demonstrate the success of the application, thus supporting the literature. By referring to the students problems they faced, and by looking at the above mentioned problems, it is evident that they are not related to the nature of the framework, but to the developers themselves. Such problems can be solved before starting any project by ensuring that developers are ready to use the framework completely, and the resources are available. Also, the current research revealed that SSDDD framework provides a better understanding of the problem situation due to the incorporation of SSM. This is in alignment with the literature, where Checkland & Scholes, (1990) have stated that SSM is a problem-solving methodology which focuses on the soft issues of a system and is applied to investigate problematic situations. Checkland & Howell (1998) have also observed the same aspect, that the use of SSM gives high clarity of the problem and issues in the system that reduces the chances of information system failure. Therefore, it is inferred that the current research is supported by the literature, where the integration of SSM solves the emerging problems of ISD.

7.2.3 Evaluating the Comparison of SSDDD with DDD and other ISD approaches

The proposed SSDDD framework has been compared to the DDD framework as an ISD development approach.

The comparison shows that for handling the perspectives of business domain modelling and implementation, the SSDDDF earned 21 points out of 24 while the DDD framework earned 16 points out of 24. The reason for this is that DDD does not use SSM techniques to model the soft perspective, and depends only the UML class diagram for modelling the other perspectives, while SSDDD uses different UML tools to model them. Thus, it may be

considered that as an ISD framework, SSDDD has improved on DDD by 12.5%. This improvement percentage fills *the second gap* in knowledge. Further improvements can be achieved in the future, which will be discussed and presented later in this chapter. In addition, the SSDDDF has introduced 'soft language' as a complement to 'ubiquitous language', which fills *the first gap* in knowledge.

7.2.2.1 Discussion of evaluation based on comparison of SSDDD with DDD and other 'IS' development approaches

SSDDD and DDD were compared to determine their capability of handling business domain perspectives. The comparison showed that SSDDD improved the capability of DDD as an ISD approach by 12.5%. This figure represents the difference between the SSDDD and DDD capability scores, which were calculated to be 87.55 and 75% respectively.

This thesis considers that, as an ISD framework, SSDDD represents a 12.5% improvement compared with DDD. This outcome fills *the second gap* in knowledge*,* 'DDD improvement', as discussed in Chapter 2.

According to Evans (2004), the structure of the ubiquitous language in DDD must be modified in a simpler manner so as to encourage the interaction for different stakeholders, especially business experts. In the present work, same has been achieved by introducing 'soft language'. SSDDD may be seen as an improvement of DDD from the following perspectives:

- The addition of SSM techniques to model the soft perspectives of the business domain, instead of depending on the UML class diagram only to model all perspectives.

- The introduction of 'soft language' in SSDDD, as a complement to 'ubiquitous language', which fills *the first gap* in knowledge.

Also, as demonstrated in chapter 6 (section 6.4), the proposed framework compared to different ISD approach such SSADM, Agile Methodologies, Multiview, and SWF framework. The brief comparison is done based on their capabilities of handling both 'soft' and 'hard' systems perspectives, using implementation patterns, and the production of a software system that has a good chance to avoid software system failure.

The proposed framework SSDDD performs better than these existing information systems development approaches determined in the literature review.

Further improvements can be achieved in the future, and such improvements are discussed and presented in the 'Future Work' section.

7.2.4 Justification of the benefits of the evaluated framework SSDDD

The evaluation suggests the evaluated approach SSDDD has delivered a number of benefits which support the evaluation criteria adopted to evaluate it. These benefits include the following:

1- Provide deep and enhanced understanding which can further help the students and developers so that they are able to apply and implement information system which can combine the requirements of business experts. The understanding of the problem is enhanced using the business domain leading to the substantial software system.

2-The applicability of the system is wide including several ranges of situations being requirement analysis for information system design.

3-Using both the techniques SSM and UML combination provides better outcomes and enhanced advantages are achieved.

4-Using this framework the whole systematic picture of business domain is understood better leading to sufficient business domain model so that the required software system can be coded.

5-The evaluating measures elaborate the applications of the framework subsequently applying it to the existing concern measures further it also identifies the extensions of the framework. There is a vast applicability of this framework in the real world development projects.

The above mentioned benefits conclude the justification of selecting such criteria to evaluate the proposed framework by highlighting the important benefits of the evaluated framework SSDDD which support the selected evaluation criteria.

7.3 Research Achievements

As stated above, the development and evaluation of the SSDDD framework has aimed to answer two research questions in order to fill the mentioned gaps in the knowledge. This process has enabled certain contributions to be made by this research. These contributions are outlined as follows:

1- The proposal of a multimethodological framework called 'Systemic Soft Domain-Driven Design (SSDDDF) to deal with both 'soft' and 'hard' business domain perspectives as an improvement of DDD. This framework can be used for information system development in an efficient manner as it addresses both, the human and technical aspects of a system.

2- The improvement of DDD as an ISD approach by an estimated percentage of 12.5%.

3- The introduction of 'soft language', as a complement to DDD's 'ubiquitous language', which consists of SSM modelling tools. The inclusion of soft language has facilitated the communication between the different stakeholders and developers, thus offering more clarity of requirements that further reduces the chances of ISD failure.

4- The demonstration and elaboration of a technique to move from SSM CPTM to UML use case diagram. As the literature revealed that the transition between SSM to UML imposes certain difficulties, the present research attempted to offer the approach for the same. However, this aspect poised itself as a complicated task and can be further improved in the future work.

5- Providing tools of implementation pattern that are compatible with the system. The tools such as Naked Objects and Trueview have been explained with screenshots that offers a better understanding of the implementation patterns.

7.4 The limitations of the evaluation framework and criteria

The adopted evaluation framework facing different limitations because of different circumstances related to this research. The time limit and the impossibility to apply the proposed framework in real business organizations which caused the evaluation to use students projects only in order to apply the framework as what done in chapter 4 and 5. In chapter 6, evaluation of SSDDDF through teaching ISD module is presented and followed with the comparison of the SSDDD with DDD and other frameworks reviewed in the literature. The available information about the existing ISD approaches, and the proposed SSDDD are used to support the evaluations done in chapter 4 and 5. To do so, the evaluation criteria is proposed to be closed to both DDD and SSDDD and to what done in

Chapter 4 , 5 and 6. The following limitations are recorded about the adapted criteria and the evaluation approach in general.

1- This evaluation framework is limited to place, person performing it, techniques for judgment, and the availability of information about the compared methods. Based on these conditions, the contributions of finding may be limited and generalized to the similar cases only.

2. Using two DDD implementation patterns i.e. Naked Objects and True View by SSDDD is another limitation as it restricts the developer to use only these two implementation patterns making the choices very limited. This limited the ability to compare the implementation results, and this may be affect the results to be not accurate. But the implementation perspective is an important part to judge the performance of the evaluated method and can't be ignored. In the other side, some developers considered this determination as an advantage since it can provide a good guidance to them.

3. The evaluated framework depends on the available information to be used through the comparison process and the accuracy of these information may be limited and will affect the acceptance of the results.

4. There is also the possibility of mismatch of the information attained using various sources.

5. The evaluation of the framework through teaching suffering from the availability of enough time to practise the different tools to provide the proper feedback and reflections. Also, the difficulties they face to convert from SSM to UML.

7.5 Limitations of SSDDDF

The SSDDD framework was proposed on the basis of gaps in the knowledge documented in the literature, and was further developed and enhanced while practising it through different illustrative ISD case studies and through teaching and practising it's tools for a larger sample of postgraduate Informatics students . However, the work has some limitations, which are detailed as follows:

- While evaluating the framework as an ISD approach, it was not possible to try it in the industry since the researcher was working as a lecturer in an academic environment and was therefore unable to get any organization to adopt the

framework and try it with one of their systems. In addition, as the methodology adopted was action research, this required the researcher to be part of the development team. It was not possible for the researcher, as a lecturer in a university, to be granted permission to do that within another organization.

- An issue was raised by the student developers' in the first evaluation stage regarding the transition from SSM CPTM to UML use case. Some of them said that this was not an easy task, as they had not practised it before, and they needed more time to do it. The 'Future Work' section will propose a solution for this.

- With regard to implementation, the use of implementation patterns like Naked Objects is a good approach, but sufficient time and resources (e.g., original software) must be available in advance, and students must try it beforehand in order to be ready for implementation. This is a problem that must be overcome, since the modelling and development are integrated parts of the framework.

- The conversion process from SSM to UML is an important part and support ISD process to be more reliable, but the conversion process must be reviewed and new approaches must be proposed to enhance it as it explained in the future work section.

The problems mentioned above have limited the contribution of this work, but they have also opened up areas for further research to be undertaken in the near future. The following section presents the future work suggested by this research.

7.6 Future Work

The above-mentioned limitations may be overcome if the following recommendations can be implemented in the future.

- Firstly, regarding real business projects, it is suggested that further attempts are made to promote the framework, through presentations to different companies, with the aim of persuading them to try using the framework. This may require some minor tailoring of the framework to fit with the organizations' requirements.

- The proposed framework can be applied as a guided learning approach for teaching with rigorous evaluations.

- Different pattern languages should be designed to handle issues with the framework. This will facilitate the job of the developers and enable learners to overcome some of the problems mentioned in the 'Limitations' section. Pattern languages usually document the successful practices of any domain, enabling them to be used by others who need to do similar work. This field is well known in architectural engineering, as it was introduced many years ago by Alexander, Ishikawa and Silverstein (1977). It was then mapped to software development patterns (Gamma, Helm, Johnson & Vlissides, 1994), and subsequently to teaching in the form of pedagogical pattern language (Bergin, 2001). The proposed pattern languages could include, but would not be limited to:

- A pattern language to facilitate UML modelling;

- Pattern language to show the conversion process from SSM CPTM to use case;

- Pedagogical pattern language to support the usage of the framework for teaching ISD.

7.7 Concluding Remarks

The present investigation aimed at developing an information system development methodology that addresses the existing issues to decrease the failure rate among information systems. For this purpose, a new framework for business domain modelling and implementation, SSDDD, has been developed. This framework considers hard and soft perspectives of the business domain by combining SSM as a guiding methodology with UML as a modelling approach and a DDD implementation pattern. A soft language has been proposed to encourage effective communication among the involved stakeholders for the purposes of understanding the dynamic system requirements. Different implementation tools such as Naked Objects and Trueview have been explained for understanding their mechanism and use. Lastly, the framework has been evaluated through different practical case studies from the academic environment, comprising undergraduate and postgraduate final projects, and by using and practising it's tools through teaching ISD module to the postgraduate students. It is inferred from the investigation that SSDDD is successful in terms of fulfilling both the hard and soft requirements and generating higher level of clarity and understanding, when compared with the previous approaches. The results achieved indicate good potential for the research to be continued in the form of further evaluation and practice in the business environment.

Bibliography

Abrahamsson, P. (2008). Agile Processes in Software Engineering and EXtreme Programming: 9Th International Conference, XP 2008, Limerick, Ireland, June 10-14, 2008: Proceedings, Springer Science & Business Media.

Adenowo, A.A. & Adenowo, B.A. (2013). Software Engineering Methodologies: A Review of the Waterfall Model and Object Oriented Approach. International Journal of Scientific & Engineering Research, Volume 4, Issue 7.

Aguilar-Saven, R. S. (2004). Business process modelling: Review and framework. International Journal of production economics 90(2): 129-149.

Ahmed, F., Capretz, L.F., Bouktif, S. & Campbell, P. (2013). Soft Skills and Software Development: A Reflection from Software Industry, *International Journal Of Information Processing And Management*, *4*(3), 171-191. http://dx.doi.org/10.4156/ijipm.vol4.issue3.17

Alexander, C., et al. (1977). A pattern language: towns, buildings, construction, Oxford University Press.

Al-Humaidan, F. and N. Rossiter (2004). Business Process Modelling with OBPM combining soft and hard approaches. Proceeding of 1st Workshop on Computer Supported Activity Coordination (CSAC), 6th International Conference on Enterprise Information Systems, Porto.

Al-Humaidan, F. M. (2006). Evaluation and development models for business processes.

Al-Mahid, M.T. & Abu-Taieh, E.M. (2006). Information Systems in Developing Countries: Reasons for Failure – Jordan, Case Study. Emerging Trends and Challenges in Information Technology Management, 1, 2.

Alter, S. (2006). The Work System Method: Connecting People, Processes, and IT for Business Results, Work System Press.

Alzubidi, N.H., Recker, J., & Bernhard, E. (2011). A Study of the Use of Business Process Modelling at Suncorp. Initial Insights. Brisbane, Australia: Business Process Management

Research Group. Retrieved from http://apromore.org/wp-content/uploads/2011/11/Suncorp-project-report-1.pdf

Ambler, S. (2002). Agile modeling: effective practices for extreme programming and the unified process, John Wiley & Sons.

Ashworth, C. and M. Goodland (1990). SSADM: A practical approach, McGraw-Hill Book Company Limited.

Avison D.E., Fitzgerald G. (1988) Information Systems development: methodologies, techniques and tools, Blackwell Scientific Publications

Avison, D. and A. Wood-Harper (1990). Multiview. An Exploration in Information Systems Development, Alfred W aller Limited.

Avison, D. E., & Fitzgerald, G. (1995). Information Systems Development: Methodologies, Techniques and Tools, Paul & Company Publishers Consortium.

Avison, D. and G. Fitzgerald (2003). Information systems development: methodologies, techniques and tools, McGraw Hill.

Avison, D. E., & Taylor, V. (1997). Information systems development methodologies: a classification according to problem situation. Journal of Information technology, 12(1), 73-81.

Barjis, J. (2008). The importance of business process modeling in software systems design. Science of Computer Programming 71(1): 73-87.

Baskerville, R. & Myers, M.D. (2004). Special Issue on Action Research in Information Systems: Making is Research Relevant to Practice— Foreword. MIS Quarterly, 28 (3), pp. 329-335.

Baskerville, R. & Wood-Harper, A. (1996). A critical perspective on action research as a method for information systems research. Journal Of Information Technology, 11(3), 235-246.

Beck, K. (2000). Extreme programming explained: embrace change, Addison-Wesley Professional.

Bell, T. E., & Thayer, T. A. (1976, October). Software requirements: Are they really a problem?. In Proceedings of the 2nd international conference on Software engineering (pp. 61-68). IEEE Computer Society Press.

Benington, H. D. (1987). Production of large computer programs. In ICSE (Vol. 87, pp. 299-310).

Bennett, S. and S. McRobb farmer, R. (2002). Object-oriented System Analysis and Design 2"'Edition. UK, McGraw Hill.

Bergin, J. (2001). A pattern language for initial course design. ACM SIGCSE Bulletin, ACM.

Bhuvaneswari, T., & Prabaharan, S. (2013). A Survey on Software Development Life Cycle Models. Journal of Computer Science and Information Technology, Vol2 (5), 263-265.

Birrell, N. D., & Ould, M. A. (1988). A practical handbook for software development. Cambridge University Press.

Boehm, B. W. (1988). A spiral model of software development and enhancement. Computer, 21(5), 61-72.

Boehm, B., & Turner, R. (2003). Balancing agility and discipline: A guide for the perplexed. Addison-Wesley Professional.

Brace, C., Bailey, A. R., & Harvey, D. C. (2006). Religion, place and space: a framework for investigating historical geographies of religious identities and communities. Progress in Human Geography, 30(1), 28-43.

Bradbury, H. and P. Reason (2003). "Action Research An Opportunity for Revitalizing Research Purpose and Practices." Qualitative social work 2(2): 155-175.

Brian, W., et al. (1999). Business Information Systems: A Process Approach, London: McGraw-Hill.

British Computer Society (2006). http://www.bcs.org/

Brown, K. (1995) Remembrance of Things Past: Layered Architectures for Smalltalk Applications. The Smalltalk Report,. 4(9): p. 4-7.

Bryman A (1984). The Debate about Quantitative and Qualitative Research: A Question of Method or Epistemology?. The British Journal of Sociology, 35(1), 75-92.

Bruner, J.S. (1966). Toward a theory of instruction, The Belknap Press of Harvard University Press, Cambridge, Massachusetts.

Burman E (1997), "Minding the Gap: Positivism, Psychology, and the Politics of Qualitative Methods" in Journal of Social Issues, Vol. 53, No.4, pp. 785-801.

Burn, J. M. and L. C. Ma (1997). Innovation in IT education-practising what we preach. Information Resources Management Journal 10(4): 16.

Bustard, D. W., Dobbin, T. J., & Carey, B. N. (1996, April). Integrating soft systems and object-oriented analysis. In Requirements Engineering, 1996., Proceedings of the Second International Conference on (pp. 52-59). IEEE.

Bustard, D. W., et al. (2000). Linking soft systems and use-case modelling through scenarios. Interacting with computers 13(1): 97-110.

Carroll, M. (1996). Peer tutoring: Can medical studies teach biochemistry? Biochemical Education 24(1): 13-15.

CCTA (1993) Applying Soft Systems Methodology to an SSADM Feasibility Study. HMSO, London.

Charvat, J. (2003). Project management methodologies: selecting, implementing, and supporting methodologies and processes for projects. John Wiley & Sons.

Checkland P.B. (1981) Systems thinking, systems practice, John Wiley, Chichester

Checkland, P. (1999). Soft Systems Methodology: a thirty year retrospective. Systems Research and Behavioral Science, Citeseer.

Checkland, P. (1999). Systems thinking. Rethinking management information systems: 45-56.

Checkland, P. and J. Poulter (2006). Learning for action: a short definitive account of soft systems methodology and its use for practitioner, teachers, and students, Wiley Chichester.

Checkland, P. and S. Holwell (1997). Information, systems and information systems: making sense of the field.

Checkland, P., and Scholes J (1990), Soft Systems Methodology in Action, John Wiley & Sons, New York,

Checkland, P., & Holwell, S. (1998). Action research: its nature and validity. Systemic Practice and Action Research, 11(1), 9-21.

Chilisa, B. and Preece, J. (2005). Research Methods for Adult Educators in Africa. UNESCO, South Africa, p 171.

Clegg, S., & McAuley, J. (2005). Conceptualising middle management in higher education: A multifaceted discourse. Journal of Higher Education Policy and Management, 27(1), 19-34.

Coad, P., et al. (1999). Feature-driven development. Java Modelling in Color with UML: 182-203.

Coad, P., et al. (1999). Java Modelling Color with Uml: Enterprise Components and Process with Cdrom, Prentice Hall PTR.

Cockburn, A. (1997). Structuring Use Cases with Goals1.

Cockburn, A. (1999). "Writing effective use cases." preparation for Addison-Wesley Longman. www. infor. uva. es/~ mlaguna/is2/materiales/BookDraft1. pdf.

Cockburn, A. and J. Highsmith (2001). Agile software development: The people factor. Computer(11): 131-133.

Coghlan, D. & Brannick, T. (2014) Doing action research in your own organization. 4th ed. London: Sage.

Cohen, D., et al. (2003). Agile software development. DACS SOAR Report(11).

Cormack, D (2000). The research process in nursing 4th ed, Blackwell Science. Oxford, p 213

Creswell, J. W. (2013b). Research design: A qualitative, quantitative and mixed method approaches. 4th edition. Sage publications.

Curtis, B., et al. (1992). Process modelling. Communications of the ACM 35(9): 75-90.

Davenport, T. h. (1993) Process innovation: Reengineering work through information technology, Harvard Business School Press, Boston, Mass.

Davenport, T. H. and J. E. Short (2003). The new industrial engineering: Information technology and business process redesign. Operations management: Critical perspectives on business and management: 97-123.

Davies, L. and P. Ledington (1988). Creativity and metaphor in soft systems methodology. Journal of applied systems analysis 15: 31-36.

Davis, G. B. (2000). Information systems conceptual foundations: looking backward and forward. In Organizational and social perspectives on information technology (pp. 61-82). Springer US.

Devaraj, S., & Kohli, R. (2003). Performance impacts of information technology: Is actual usage the missing link?. Management science, 49(3), 273-289.

Dhillon, G. S. and J. Ward (2002). Chaos theory as a framework for studying information systems. Advanced topics in information resources management 2: 320-338.

Dick, B. (2002). Soft systems methodology. Session 13 of Areol - action research and evaluation [online] available at: <http://www.scu.edu.au/schools/gcm/ar/areol/areol-session13.html#a_s13_7s> [accessed on 18th June 2010]

Dogan, H. and M. Henshaw (2010). Transition from soft systems to an enterprise knowledge management architecture. International Conference on Contemporary Ergonomics and Human Factors, April.

Donaldson, A. J. M., & Jenkins, J. O. (2001). Systems failures: An approach to building a coping strategy. In Software Engineering Education and Training, 2001. Proceedings. 14th Conference on (pp. 187-190). IEEE.

Donnelly, J., & Trochim, W. (2007). The research methods knowledge base. Ohio: Atomic Dog Publishing.

Erikksonn, H. and M. Penker (2000). UML business process modelling at work, John Wiley and Sons, New York.

Ertugrul, A. M. and O. Demirors (2015). An exploratory study on role-based collaborative business process modeling approaches. Proceedings of the 7th International Conference on Subject-Oriented Business Process Management, ACM.

Evans, E. (2004). Domain-driven design: tackling complexity in the heart of software, Addison-Wesley Professional.

Ewusi-Mensah, K. (2003). Software development failures. Mit Press.

Feiler, P. H. and W. S. Humphrey (1993). Software process development and enactment: Concepts and definitions. Software Process, 1993. Continuous Software Process Improvement, Second International Conference on the, IEEE.

Fenning, R., Dogan, H and K. Phalp (2014). Applicability of SSM and UML for Designing a Search Application for the British Broadcasting Corporation (BBC). Enterprise, Business-Process and Information Systems Modeling, Springer: 472-486.

Fondas, N. (1993). Process Innovation: Reengineering Work through Information Technology. The Academy of Management Executive 7(2): 100-103.

Fowler, M. (2004). UML distilled: a brief guide to the standard object modeling language, Addison-Wesley Professional.

Fredrick Tylor (1911) The Principles of Scientific Management, Monograph published by Harper & Brothers. P 114.

Gamma, E., et al. (1994). Design patterns: elements of reusable object-oriented software, Pearson Education.

Gardner, H. (1993). Multiple intelligences: The theory in practice, Basic books.

Georgakopoulos, D., et al. (1995). "An overview of workflow management: From process modeling to workflow automation infrastructure." Distributed and parallel Databases 3(2): 119-153.

Giddens, A. (1984). The constitution of society: Outline of the theory of structuration, Univ of California Press.

Morgan, G. A., Gliner, J. A., & Harmon, R. J. (2006). *Understanding and evaluating research in applied and clinical settings*. Psychology Press.

Goodlad, S. and B. Hirst (1989). Peer Tutoring. A Guide to Learning by Teaching, ERIC.

Gorgone, J., et al. (2003). IS 2002 model curriculum and guidelines for undergraduate degree programs in information systems. Communications of the Association for Information Systems 11(1): 1.

Goyal, D. (2012). Business alignment and critical success factors in information systems implementation: an empirical analysis of selected Indian organisations. International Journal of Business Information Systems 10(4): 397-416.

Gray, R., Kouhy, R. and Lavers, S. (1995). Methodological themes: constructing a research database of social and environmental reporting by UK companies. Accounting, Auditing and Accountability Journal, Vol. 8 No 2, pp. 78-101

Greenwood, D.J. & Levin, M. (2007) Introduction to action research. 2nd ed. Thousand Oaks, California: Sage.

Grinnel, R. M. and Unrau, Y. A. (2008). Social Work research and evaluation: Foundations of evidence based practice. Oxford University Press: New York

Gummesson, E. (2000). Qualitative methods in management research, Sage.

Gupta M and Gupta D (2011), Research Methodology, PHI Learning Private Limited, New Delhi, p 32.

Gupta, J. and R. Wachter (1998). A capstone course in the information systems curriculum. International Journal of Information Management 18(6): 427-441.

Hammer, M. (1990). Reengineering work: don't automate, obliterate." Harvard business review 68(4): 104-112.

Hammer, M. and J. Champy (2009). Reengineering the Corporation: Manifesto for Business Revolution, A, Zondervan.

Harry, M. (1994). Information Systems in Business, Pitman, London.

Hasan, H. (2003). Information Systems Development as a Research Method. *AJIS*, *11*(1). http://dx.doi.org/10.3127/ajis.v11i1.142\

Hendricks, K. B., Singhal, V. R., & Stratman, J. K. (2007). The impact of enterprise systems on corporate performance: A study of ERP, SCM, and CRM system implementations. Journal of Operations Management, 25(1), 65-82.

Highsmith, J. (2013). Adaptive software development: a collaborative approach to managing complex systems, Addison-Wesley.

Highsmith, J. A. (2002). Agile software development ecosystems, Addison-Wesley Professional.

Highsmith, J. and A. Cockburn (2001). Agile software development: The business of innovation. Computer 34(9): 120-127.

Hoffman, K. and P. Eugster (2009). Cooperative aspect-oriented programming. Science of Computer Programming 74(5): 333-354.

Holwell, S. and P. Checkland (1998). Information, systems and information systems, Chichester, England: John Wiley.

Höysniemi, J., et al. (2003). Using peer tutoring in evaluating the usability of a physically interactive computer game with children. Interacting with computers 15(2): 203-225.

Jacobson, I., Booch, G., Rumbaugh, J., Rumbaugh, J., & Booch, G. (1999). The unified software development process (Vol. 1). Reading: Addison-wesley.

J Martin, C Finkelstein; (1981). Information engineering, Vols 1 and 2, Prentice-Hall.

Jacope Romei (2009). Agile development and Domain Driven Design. Online: http://www.slideshare.net/jakuza78/sviluppo-agile-e-domain-driven-design Retrieved June, 2015.

Jak Charlton (2010). A practical guide to domain driven design. Online: -thinkddd.com.

Jeff De Luca (1997). Feature-Driven Development.

Johansson, H. J., et al. (1993). Business process reengineering: Breakpoint strategies for market dominance, Wiley Chichester.

Johnson, B. and Christensen, L. B. (2010). Educational Research: Quantitative, Qualitative, and Mixed Approaches. London: SAGE Publications.

Jones, D. and A. Newman (2002). A constructivist-based tool for operating systems education. Proc. EdMedia.

Joseph Ucizi Mtenje (2010). An online application to support a peer-tutoring system for undergraduate programming modules. Msc dissertation, Informatics Department, University of Huddersfield.

Jupp, V. (ed.) (2006) The Sage Dictionary of Social Research Methods, London, Sage.

Katina, P. F., Keating, C. B., & Ra'ed, M. J. (2014). System requirements engineering in complex situations. Requirements Engineering, 19(1), 45-62.

Kaur, B.P. & Aggrawal, H. (2013). CRITICAL FAILURE FACTORS IN INFORMATION SYSTEM: AN EXPLORATORY REVIEW. Journal of Global Research in Computer Science, 4(1).

Kemmis, S., & McTaggart, R. (2005). Participatory Action Research: Communicative Action and the Public Sphere. Sage Publications Ltd.

Kettinger, W. J., et al. (1997). Business process change: a study of methodologies, techniques, and tools. MIS quarterly: 55-80.

Keys, P. and M. Roberts (1991). Information system development and soft systems thinking: Towards an improved methodology. Systems Thinking in Europe, Springer: 577-581.

Kingston, J. K. (1995). Modelling business processes using the soft systems approach, University of Edinburgh, Artificial Intelligence Applications Institute.

Kivuva, T. (2012). Challenges in development and implementation of Information systems in ad hoc landing and Over-flight clearances in the Kenyan airspace (Doctoral dissertation, University of Nairobi).

Kock, N., et al. (2002). Can Action Research be Successfully Used in Information Systems Doctoral Research? a Panel Discussion. Informing Science June Q 7.

Koshy, E., et al. (2010). Action research in healthcare, Sage.

Kruchten, P. (2004). The rational unified process: an introduction, Addison-Wesley Professional.

Laddad, R. (2009). Aspect in action: enterprise AOP with spring applications, Manning Publications Co.

Lai, L. S.-l. (2000). An integration of systems science methods and object-oriented analysis for determining organizational information requirements. Systems Research and Behavioral Science 17(2): 205.

Lane, C. and K. Galvin (1999). Methods for Transitioning from Soft Systems Methodology (SSM) Models to object oriented analysis (OOA), developed to support the Army Operational Architecture (AOA) and an Example of its Application. Proceedings of the 1999 Command and Control Research and Technology Symposium.

Laudon, K., Laudon, J. (2009). Essentials of management information systems (8th ed.). Upper Saddle River, NJ: Pearson Prentice Hall.

Lavallée, M., & Robillard, P. N. (2015, May). Why good developers write bad code: An observational case study of the impacts of organizational factors on software quality. In Proceedings of the 37th International Conference on Software Engineering-Volume 1 (pp. 677-687). IEEE Press.

Lee, D. M., et al. (1995). Critical skills and knowledge requirements of IS professionals: a joint academic/industry investigation. MIS quarterly: 313-340.

Lewis, P. (1995). Applying soft systems methodology to an SSADM feasibility study: CCTA. Systems Practice 8(3): 337-340.

Lindsay, A., et al. (2003). Business processes—attempts to find a definition. Information and software technology 45(15): 1015-1019.

Lonchamp, J. (1993). A structured conceptual and terminological framework for software process engineering. ICSP, Citeseer.

Loos, S., et al. (2005). Three perspectives on peer tutoring for CS1. Proceedings of the Midwest Celebration of Women in Computing conference, http://www. cs. indiana. edu/cgi-pub/midwic/papers/uploads/loos. pdf.

Loucopoulos, P. (2003). The S3 (Strategy-Service-Support) Framework for Business Process Modelling. CAiSE Workshops, Citeseer.

Lucky, E. O. I., Adegoke, O., & Othman, N. (2014). Project management challenges and difficulties: A case study of information system development. International Postgraduate Business Journal, 6(1), 99-133.

Lunn, K. (2003). Software Development with UML, Palgrave Macmillan.

Lyytinen, K. and D. Robey (1999). Learning failure in information systems development. Information Systems Journal 9(2): 85-101.

Manalil, J. (2011). Rational Unified Process. Maidenhead, UK: Open University Press.

Mansell, G. (1991). Action research in information systems development. Information Systems Journal, 1(1), 29-40. http://dx.doi.org/10.1111/j.1365-2575.1991.tb00025.x

Matković, P., & Tumbas, P. (2010). A Comparative Overview of the Evolution of Software Development Models. International Journal of Industrial Engineering and Management (IJIEM), ISSN, 2217-2661.

Maxcy, S. J. (2003). Pragmatic threads in mixed methods research in the social sciences: The search for multiple modes of inquiry and the end of the philosophy of formalism. Handbook of mixed methods in social and behavioral research, 51-89.

McPherson, M. and M. B. Nunes (2004). Developing innovation in online learning: an action research framework, Psychology Press.

Medina-Mora, R., et al. (1992). The action workflow approach to workflow management technology. Proceedings of the 1992 ACM conference on Computer-supported cooperative work, ACM.

Melville, N., Kraemer, K., & Gurbaxani, V. (2004). Review: Information technology and organizational performance: An integrative model of IT business value. MIS quarterly, 28(2), 283-322.

Meyer, J. (2000) 'Using qualitative methods in health related action research', British Medical Journal, 320: pp. 178–181. [Online]. Available from: http://dx.doi.org.ezproxy.liv.ac.uk/10.1136/bmj.320.7228.178 (Accessed: 24 February 2015).

Michiel Uithol (2009). Security in Domain-Driven Design. Master dissertation, University of Twente.

Mihailescu, D. and M. Mihailescu (2010). A realist conceptualization for studying information system development methodology. The 1st International Conference on Information Management and Evaluation (ICIME 2010), Acad Conferences Ltd.

Mike Bennet (2007). Domain Modelling Tools. http://www.hypermedia.co.uk. Retrieved July,2015.

Miles, R., (1992) Combining "hard" and "soft" systems practice: grafting and embedding revisited. Systemist 14 (2) 62-66.

Miliszewska, I. and G. Tan (2007). Befriending computer programming: A proposed approach to teaching introductory programming." Informing Science: International Journal of an Emerging Transdiscipline 4(1): 277-289.

Mingers, J. (1988). "Comparing conceptual models and data flow diagrams." The Computer Journal 31(4): 376-379.

Mingers, J. (2000). Variety is the spice of life: combining soft and hard OR/MS methods. International Transactions in Operational Research, 7(6), 673-691.

Mingers, J., & Brocklesby, J. (1997). Multimethodology: towards a framework for mixing methodologies. Omega, 25(5), 489-509.

Mingers, J. (2001). "Combining IS research methods: towards a pluralist methodology." Information systems research 12(3): 240-259.

Mingers, J. C. (1995). Information and meaning: foundations for an intersubjective account. Information Systems Journal, 5(4), 285-306.

Mingers, J., & Taylor, S. (1992). The use of soft systems methodology in practice. Journal of the Operational Research Society, 321-332.

Mishra, S. (2004). Visual Modelling & Unified Modelling Language. Online: Accessed on October, 2010 from : http://www2.informatik.hu-berlin.de/~hs/Lehre/2004-WSSWQS/20050107UML.PPT

Morgan, G. A., Gliner, J. A., & Harmon, R. J. (2006). Understanding and evaluating research in applied and clinical settings. Psychology Press.

Munassar, N.M. & Govardhan, A. (2010). A Comparison Between Five Models Of Software Engineering. IJCSI International Journal of Computer Science Issues, 7 (5)

Mynard, J. and I. Almarzouqi (2006). "Investigating peer tutoring." Elt Journal 60(1): 13-22.

Naked Objects (2010) Naked Objects MVC - Product description [online] available at: <http://nakedobjects.net/product/product_intro.shtml> [accessed on 15th August 2010]

Object Management Group (OMG) (2005) Introduction to OMG's Unified Modeling Language™ (UML®) [online] available at: <http://www.omg.org/gettingstarted/what_is_uml.htm

Oliver I., Kent, S. (2009) Validation of Object Oriented Models using Animation [online] available at: <http://kar.kent.ac.uk/21768/1/validation_of_object-oriented_oliver.pdf> accessed on 24th June 2010.

Oldfield, P., (2002). Domain Modelling, Appropriate Process Group. www.aptprocess.com.

Retrieved on June,2015.

Oqvist, J. (2011). Becoming More Agile With Domain-Driven Design.

Orb, A., Eisenhauer, L., & Wynaden, D. (2001). Ethics in qualitative research. Journal of nursing scholarship, 33(1), 93-96.

Ould, M. A. and M. Ould (1995). Business Processes: Modelling and analysis for re-engineering and improvement, Wiley Chichester.

Parkin, P. (2009). Managing change in healthcare using action research. London: Plagrave.

Patel, N. V. (1995). Application of soft systems methodology to the real world process of teaching and learning. International Journal of Educational Management 9(1): 13-23.

Patton, M. Q. (2002). Qualitative Research and Evaluation Methods, SAGE Publications, London.

Pawson, R. and R. Matthews (2002). Naked objects. Companion of the 17th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, ACM.

Platt, D. and D. Blockley (1994). Process modelling in civil engineering design. Design Studies 15(3): 317-331.

Pressman, R. S. (1994). History of Automated Guided Vehicles.

Prior, R. (1992). Linking SSM and IS development. Systemist 14(3): 128ą132.

Reason, P. and H. Bradbury (2001). Handbook of action research: Participative inquiry and practice, Sage.

Reason, P. and H. Bradbury (2005). Handbook of action research: concise paperback edition, Sage.

Reason, P., & Bradbury, H. (2008). Handbook of Action Research: Participative inquiry and practice 2nd edition. London: Sage Publications.

Robertson, I., et al. (1999). Business information systems: a process approach, McGraw-Hill.

Rose, J. (2002). Interaction, transformation and information systems development-an extended application of Soft Systems Methodology. Information Technology & People 15(3): 242-268.

Rosenberg, D. and M. Stephens (2007). Robustness Analysis. Use Case Driven Object Modeling with UML: Theory and Practice: 101-142.

Royce, W. W. (1970). Managing the development of large software systems. In proceedings of IEEE WESCON (Vol. 26, No. 8, pp. 328-338).

Ruparelia, N.B. (2010). Software Development Lifecycle Models. ACM SIGSOFT, 35 (3)

Ryan, H. W. (1999). Managing Development in the Era of Large Complex Systems. IS Management 16(2): 89-91.

Sabherwal, R., Jeyaraj, A., & Chowa, C. (2006). Information system success: individual and organizational determinants. Management science, 52(12), 1849-1864.

Sauser, B. J., Reilly, R. R., & Shenhar, A. J. (2009). Why projects fail? How contingency theory can provide new insights–A comparative analysis of NASA's Mars Climate Orbiter loss. International Journal of Project Management, 27(7), 665-679.

Salahat, M., et al. (2009). "The Application of A systemic Soft Domain Driven Design Framework." World Academy of Science, Engineering and Technology 57(86): 476-486.

Saraj, Din (2009). MSc Dissertation, University of Huddersfield, 2009.

Saunders, M. and P. Lewis (1945). "& Thornhill, A. 2012." Research methods for business students.

Scott, M. F. K. and M. Fowler (2000). UML Distilled: A Brief Guide to the Standard Object Modeling Language, Addison Wesley.

Sensuse, D. I. and A. Ramadhan (2012). The Relationships of Soft Systems Methodology (SSM), Business Process Modeling and e-Government. IJACSA) International Journal of Advanced Computer Science and Applications 3(1).

Sewchurran, K. and D. Petkov (2007). A systemic framework for business process modelling combining soft systems methodology and UML. Information Resources Management Journal 20(3): 46.

Shenton, A.K (2004). Strategies for ensuring trustworthiness in qualitative research projects. Education for Information 22 (2004) 63–75.

Shoval, P., et al. (2006). Class Diagrams and Use Cases–Experimental Examination of the Preferred Order of Modeling. Proceedings of CAiSE 2006 workshop on Exploring Modeling Methods for System Analysis and Design (EMMSAD, Citeseer.

Silverman, D. (2013). *Doing qualitative research: A practical handbook*. 4th edition. SAGE Publications Limited.

Smeds, R. (1987). The role of computerized information systems in the development of organizational structure. International Studies of Management & Organization: 90-104.

Somekh, B. (2006). Action research: A methodology for change and development.

Srini Penchikala (2008). Domain Driven Design and Development in Practice. Online article: http://www.infoq.com/articles/ddd-in-practice

Stamouli, I., et al. (2004). Establishing structured support for programming students. Frontiers in Education, 2004. FIE 2004. 34th Annual, IEEE.

Štolfa, S. and I. Vondrák (2008). Mapping from business processes to requirements specification. Retrieved on 7th Aug.

Strong, D. M., & Volkoff, O. (2010). Understanding Organization—Enterprise System Fit: A Path to Theorizing the Information Technology Artifact. MIS quarterly, 731-756.

Stowell, F. A. and D. West (1995). Client-Led Design: A Systemic Approach to Information System Definition, McGraw-Hill, Inc.

Tashakkori & C Teddlie (Eds.), Handbook of mixed methods in social and behavioral research (pp. 51-89). Thousand Oaks, CA: Sage.

Thomas R M (2003), Blending qualitative & quantitative research methods in theses and dissertations, Corwin Press, California.

Tutorialpoint.com: http://www.tutorialspoint.com/uml/ Retreived July, 2015.

Uithol, M. (2008). Security in domain-driven design.

Ulin, P. R., Robinson, E. T., & Tolley, E. E. (2012). *Qualitative methods in public health: a field guide for applied research*. John Wiley & Sons.

van Dillen, E. (2007). Domain Driven Design in de Praktijk.

Van De Kar, E. A. M., & Verbraeck, A. (Eds.). (2008). Designing Mobile Service Systems-Revised Second Edition (Vol. 2). IOS press.

Volkoff, O., Strong, D. M., & Elmes, M. B. (2007). Technological embeddedness and organizational change. Organization Science, 18(5), 832-848.

Vonk R. (1990) Prototyping: The effective use of CASE technology, Prentice-Hall, London

Wade, S. and J. Hopkins (2002). A Framework for Incorporating Systems Thinking into Object Oriented Design. Seventh CAiSE/IFIP8.

Wade, S., et al. (2012). A Scaffolded Approach to Teaching Information Systems Design. Innovation in Teaching and Learning in Information and Computer Sciences 11(1): 56-70.

Wang, Q., et al. (2014). Research on Domain Driven Design Based Domain Platform Architecture. 2014 International Conference on Mechatronics, Electronic, Industrial and Control Engineering (MEIC-14), Atlantis Press.

Warboys, B., et al. Business Information Systems: A Process Approach. 1999, McGraw-Hill.

Williams B. (2005). Soft Systems Methodology. [online] available at: [accessed on 18th June 2010].

Wilson, B. (2001). Soft systems methodology: conceptual model building and its contribution."

Wilson, Brian (1990), System: Concept, methodologies and applications. John Wiley, New York

Winklhofer, H. (2002). "Information systems project management during organizational change." Engineering Management Journal 14(2): 33-37.

Wysocki, R. R. (2009). Effective Project Management: Traditional, Agile, Extreme (5th ed.). Indianapolis, IN: Wiley.

Wood-Harper, T. (1985). Research methods in information systems: using action research. Research methods in information systems: 169-191.

Xia, W. & Lee, G. (2005). Complexity of Information Systems Development Projects: Conceptualization and Measurement Development. Journal of Management Information Systems, 22 (1), pp. 45-83

Xiaohui, H. (2006). Improving teaching in Computer Programming by adopting student-centred learning strategies. The China Papers 6: 46-51.

Xiaohui, H. (2006). Improving teaching in Computer Programming by adopting student-centred learning strategies. The China Papers 6: 46-51.

Yusuf, L., Folorunso, O., Akinwale, A., & Adejumobi, I. (2011). Visualizing and Assessing a Compositional Approach to Service-Oriented Business Process Design Using Unified Modelling Language (UML). Computer And Information Science, 4(3).

Yourdon, E. (1989). *Modern Structured Analysis*, Yourdon Press Computing Series.

Zwass, V.(2016). Information Systems. Retrieved from : Encyclopaedia Britannica (https://global.britannica.com/topic/information-system)

Appendices

Appendix 1

Feedback Questionnaire

University of Huddersfield- Informatics Department

The module: "Methods and Modeling" for MSc students

Part One: General Information:

Name(Optional):---------- Gender:----------

Qualification: ----------Major:----------

Age:----------

Part Two: Tools and Techniques

This module has been structured around a framework of techniques that guide you through the systems development process from requirements analysis to system implementation.

The framework combines techniques from SSM, UML, and various implementation patterns for business system development. We want to continue to develop this framework for use in teaching and "real world" software development. You can help us to fine-tune the framework by answering a few simple questions.

Answer the following questions based on this briefing and the knowledge you gained from the module.

1-Understanding and practicing Soft System Methodology Tools:

Choose (5=strongly agree, 4=Agree, 3=don't know, 2=Disagree, 1=Strongly Disagree)

1- I found the tools of SSM were easy to use :

(1 2 3 4 5)

2- I can see how SSM tools would help me to understand the logic of business processes

(1 2 3 4 5)

3- I can see how SSM tools would help me to understand customer requirements

(1 2 3 4 5)

4- I can see how SSM tools could facilitate communication between business experts and developers

(1 2 3 4 5)

5- I found it easy to understand and communicate with my team using SSM techniques

(1 2 3 4 5)

6- I can see how an SSM Rich Picture can provide a comprehensive overview of a business system

(1 2 3 4 5)

7- I can see that SSM Root definition technique depicts the required system objectives

(1 2 3 4 5)

8- I am confident that I could use SSM Conceptual Models to depict the detailed logic of business processes.

(1 2 3 4 5)

9- I can see how SSM conceptual models represent the business domain processes

(1 2 3 4 5)

10- I am confident that I could use SSM techniques to identify the user requirements

(1 2 3 4 5)

2- Understanding and practicing UML Tools:

For the following questions: Choose (5=strongly agree, 4=Agree, 3=don't know, 2=Disagree, 1=Strongly Disagree)

11- 1- I found that UML is easy to use for modeling business processes.

(12 3 4 5)

12-2- I can see how Use Case diagram can be used to represent system processes.

13-(1 2 3 4 5)

3- I am confident that UML Use Cases are good tools for business process modeling

14- (1 2 3 4 5)

15- 4- I found it easy to extract Use Cases from the SSM Conceptual model

16- (1 2 3 4 5)

17- 5- I found it easy to draw a sequence diagram based on each use case.

18- (1 2 3 4 5)

19- 6- I found it easy to draw the Class Diagram based on the sequence diagrams.

20- (1 2 3 4 5)

21- 7- I can see that UML Class Diagram represents the domain model of the investigated system.

22- (1 2 3 4 5)

23- 8- I understand how code can be generated from the domain model (Class diagram).

24- 2 3 4 5)

9- Which UML Diagram you believe is the most important one for business domain modeling among other UML diagrams, rank them from the highest to the lowest using (1=Most important, 2=important,3=average,4=less than average, not important). Please put (√) in the cell you believe it's suitable.

Diagram/Importance Degree	5=Most Important	4=Important	3=Average	2=less than average	1=not important
Use Case Diagram					
Class Diagram					
Activity Diagram					
Sequence Diagram					
State Chart					
Collaboration Diagram					

1- Understanding and practicing linking between SSM and UML:

SSM provided a general understanding and conceptual modelling of the problematic situation in the business domain. The output generated by SSM will be used to model, design, and implement the required system. Based on the work you done in the course which includes moving from SSM Conceptual model to UML Use Case diagram, please answer the following questions:

1- I found the transition from Conceptual Models to Use Case Models is an easy process

(1 2 3 4 5)

2- I found that some of the activities in the Conceptual Model did not map directly to use cases.

(1 2 3 4 5)

3- I can see that the resultant use cases represent the key activities of the conceptual model

(1 2 3 4 5)

4- I found that the adapted method for transition is easy to use and practice

(1 2 3 4 5)

5- I'm confident that I can depend on the resultant use cases to draw other diagrams like sequence and class diagrams

(1 2 3 4 5)

6- I found it's useful to use SSM at the beginning to investigate the business domain and to move to UML and implementation

(1 2 3 4 5)

2- Understanding and practicing the Implementation Pattern:

Naked Objects, **TrueView, BlueJ** or other implementation patterns satisfied the philosophy of Domain Driven Design recommended to be used for implementation. The proposed framework will not deal more with the implementation part and will continue the same as DDD. If you used any of the above mentioned Patterns for implementation, please answer these questions:

1- I found the implementation pattern is an easy to adapt and use for implementation(Name of pattern:----------)

(1 2 3 4 5)

2- I found moving from Domain model (class diagram) to code is easy and not complicated

(1 2 3 4 5)

3- I found the implementation pattern easy to represent the domain model processes in code.

(1 2 3 4 5)

4- The interfaces generated by the implementation pattern are easy to use.

(1 2 3 4 5)

3- Understanding and practicing the integration of SDDD framework components:
Domain Driven Design Approach (Eric Evan, 2004) is an approach adapted to develop this framework. The developed framework expected to do some improvement in the early stages of DDD. SSSM added for investigating and modeling the business domain. It is expected to facilitate the communication between different stakeholders. Based on that new layer added to DDD (soft layer) represented by SSM. Based on this brief answer the following questions:

1- I found that integrating all the above tools in one development framework helped me to do the required project Easley

(1 2 3 4 5)

2- I'm confident that this framework can be used to develop a complete software support system

(1 2 3 4 5)

3- I'm confident that the whole systems components (soft and hard) can be

investigated, modelled, and implemented using this framework.

(2 3 4 5)

4-I found that this framework helped me to see an integrated picture of the required

system in the project

(1 2 3 4 5)

Appendix 2

Use cases porformas for PTS (undergraduate)

ID No.:	1		
Use-case Name:	Create/ adjust a peer tutor		
Description:	To be assigned to a tutoring session		
Trigger:	Registrar creates/ adjusts a peer tutor		
Trigger Type:	External		
Major Inputs: Description: Tutor identification info Knowledge area(s)	Source: Registrar Registrar	**Major Outputs:** Description: Tutor Object	Destination: Tutor Object Store (OS)

Table Appendix 2-1: Use Case for Creating/ Adjusting a Peer Tutor

ID No.:	2		
Use-case Name:	Create/ adjust a peer tutee		
Description:	To be enrolled in a tutoring session		
Trigger:	Registrar creates/ adjusts a peer tutee		
Trigger Type:	External		
Major Inputs: Description: Tutee identification info	Source: Registrar	**Major Outputs:** Description: Tutee Object	Destination: Tutee OS

Table Appendix 2-2: Use Case for Creating/ Adjusting a Peer Tutee

ID No.:	3		
Use-case Name:	Create/ adjust a peer tutoring session		
Description:	None		
Trigger:	Registrar creates/ adjusts a peer tutoring session		
Trigger Type:	External		
Major Inputs: Description: Session description Tutor object Tutee object(s)	Source: Registrar Tutor OS Tutee OS	**Major Outputs:** Description: Session Object	Destination: Session OS

Table Appendix 2-3: Use Case for Creating/ Adjusting a Peer Tutoring Session

ID No.:	4		
Use-case Name:	Insert a tutor attendance record		
Description:	If session is due, accept, otherwise reject Call use-cases 5 & 6 either way		
Trigger:	Registrar inserts/ adjusts a tutor attendance record		
Trigger Type:	External		
Major Inputs: Description: Session object DueOrNot Tutor object	Source: Session OS Session object Session object	Major Outputs: Description: Attendance Record Object	Destination: Attendance Record Object Store (OS)

Table Appendix 2- 4: Use Case for Inserting a Tutor Attendance Record

ID No.:	5		
Use-case Name:	Calculate amount receivable to tutor		
Description:	Check whether tutor's attendance is present or not, and calculate correspondingly		
Trigger:	Use-case 4		
Trigger Type:	Temporal		
Major Inputs: Description: Fee amount Session object Tutor object	Source: (Fixed per hour) Session OS Session object	Major Outputs: Description: Amount Receivable Object	Destination: Amount Rec. OS

Table Appendix 2-5: Use Case for Calculating Amount Receivable by Tutor

Appendix 3

PTS implementation using naked objects

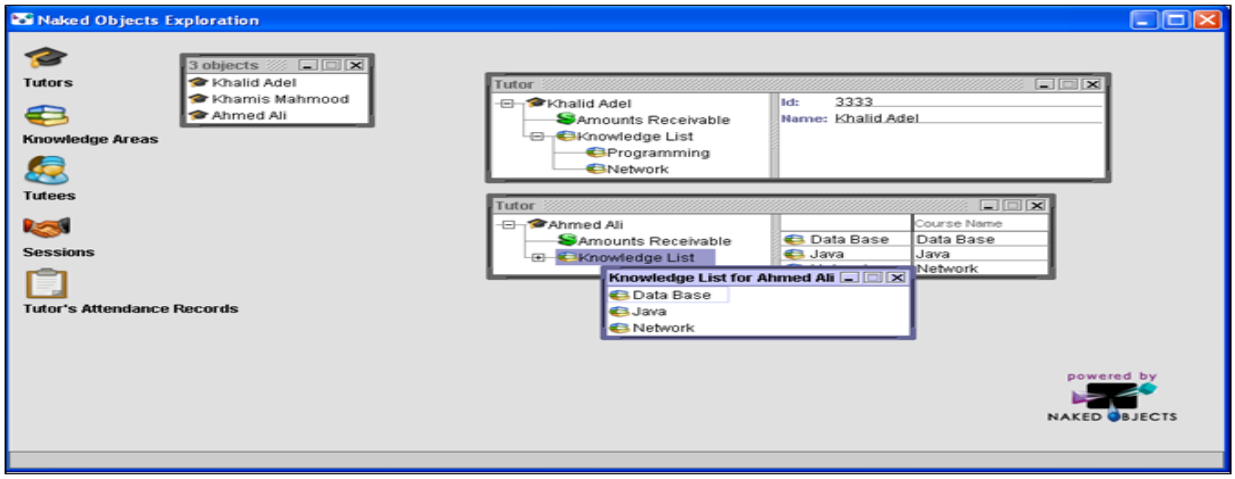


Figure Appendix 3- 1: PTS Implementation Screen Shot

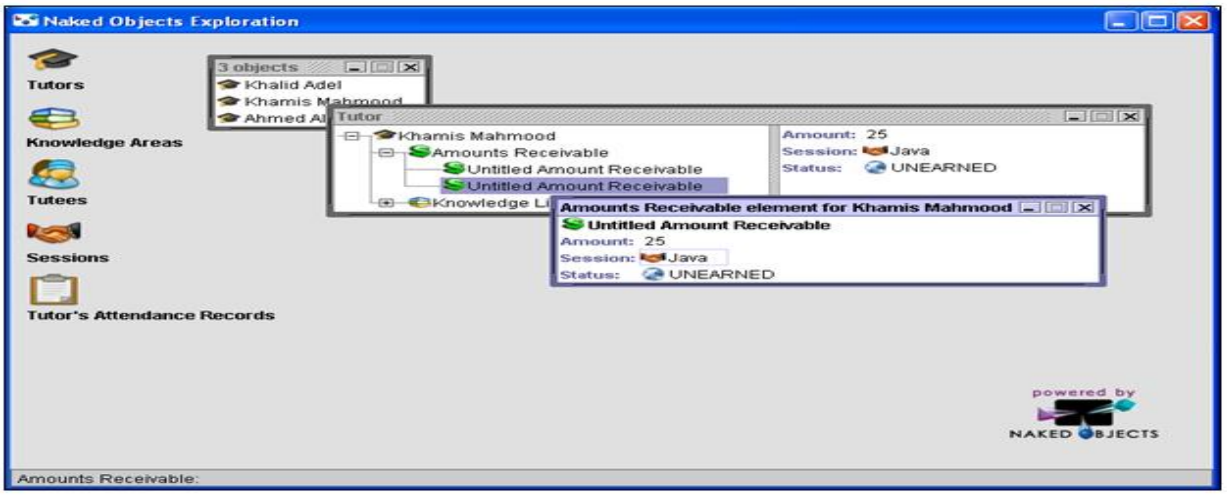


Figure Appendix 3- 2: PTS Implementation Screen Shot

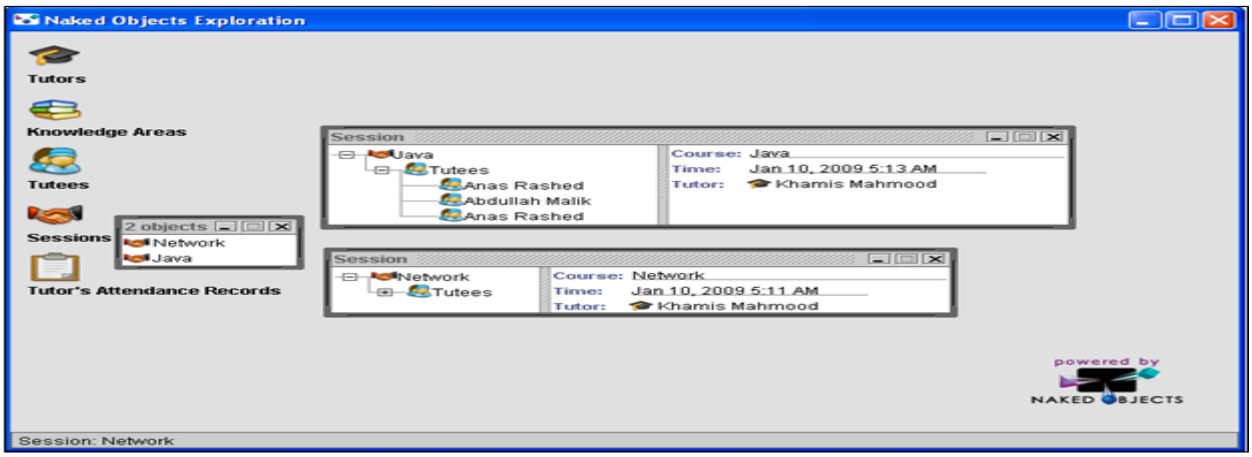


Figure Appendix 3- 3: PTS Implementation Screen Shot

Appendix 4

Activity diagrams of SAS

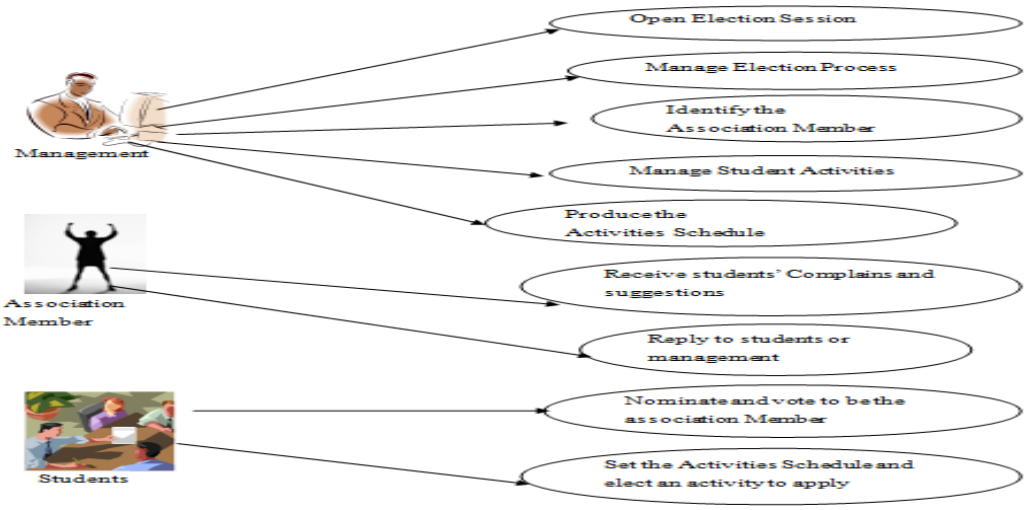


Figure Appendix 4-1: Activity Diagram for Management, Association and Students

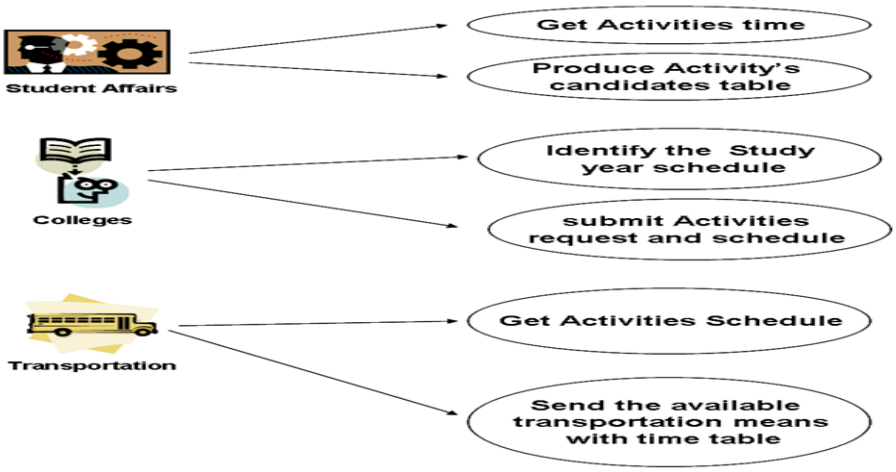


Figure Appendix 4- 2: Activity Diagram for Student Affairs, Colleges and Transportation

Figure Appendix 4- 3: Activity diagram for the election process

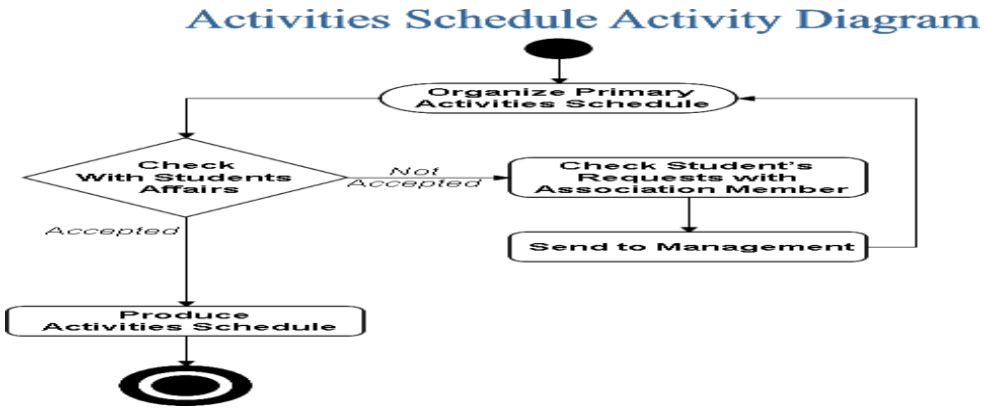


Figure Appendix 4- 4: Activity Diagram for Preparing Activities Schedule

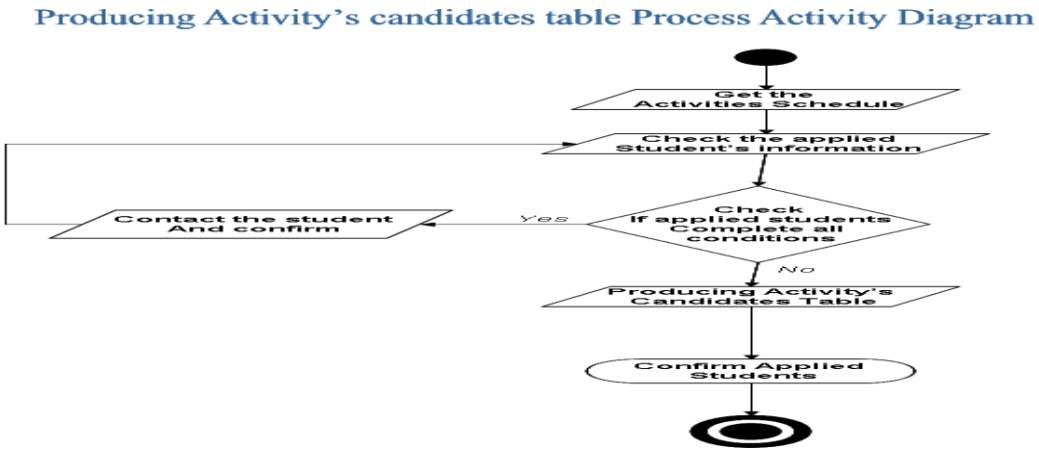


Figure Appendix 4- 5: Activity Diagram for Preparing Candidate Schedule

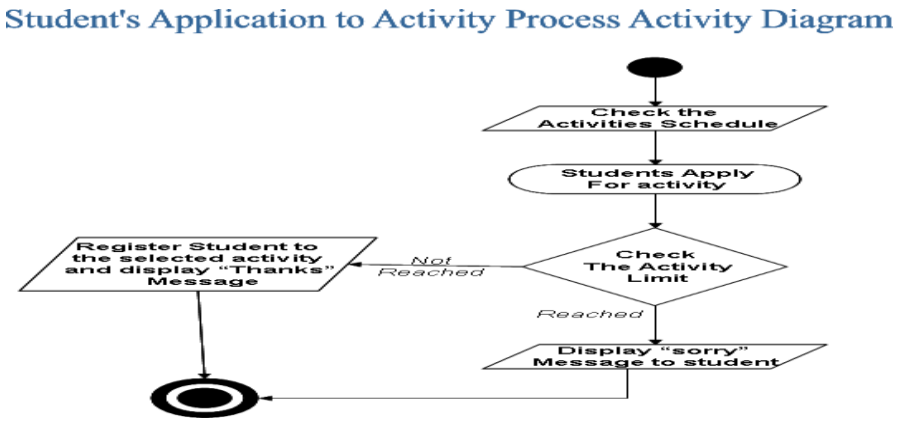


Figure Appendix 5- 6: Activity Diagram for Preparing Student Application

Appendix 5

SAS implementation using naked objects

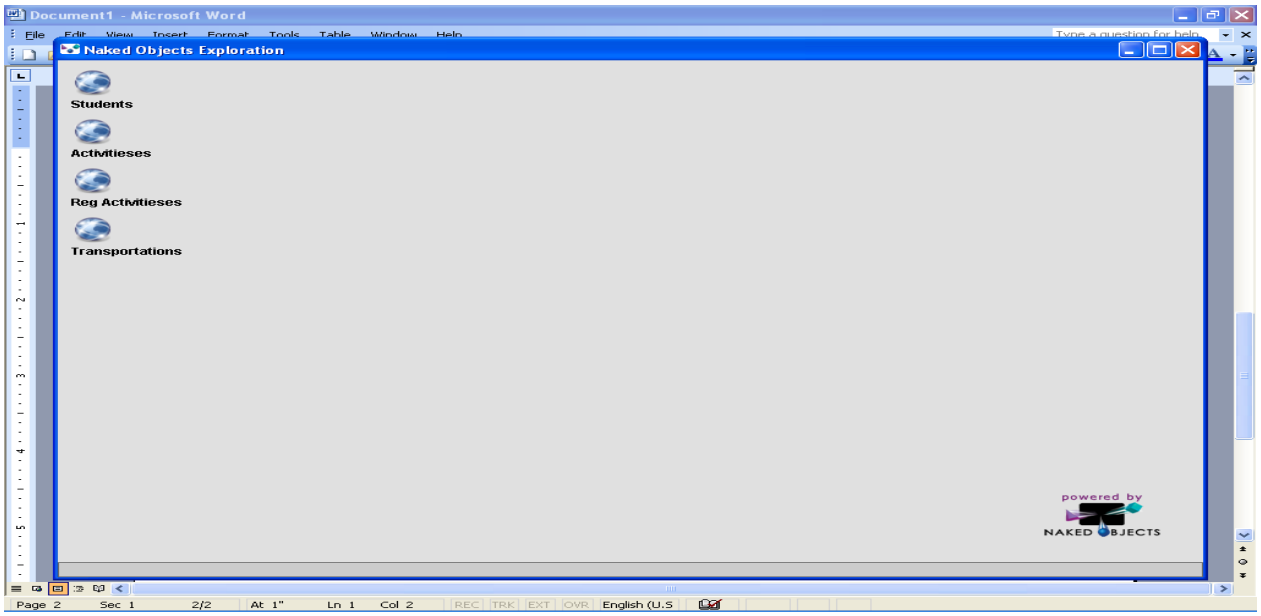


Figure Appendix 5- 1: Main Menu of SAS Software Screen Shot

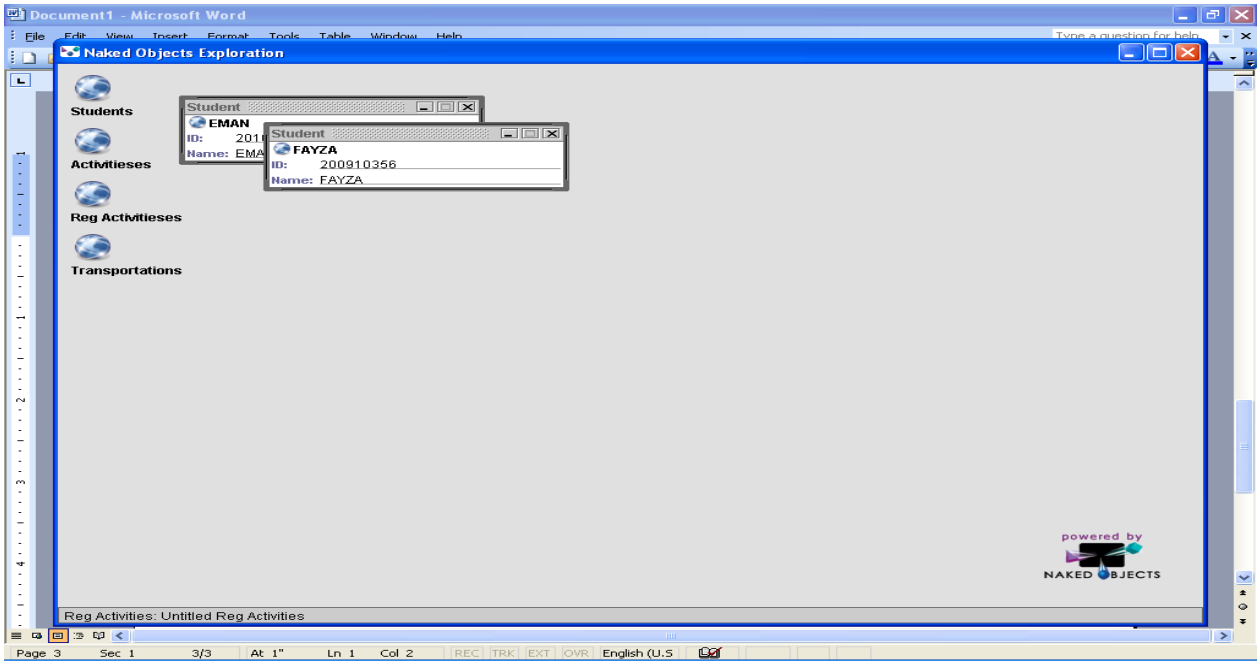


Figure Appendix 5- 2: Data Entry Screen Shot

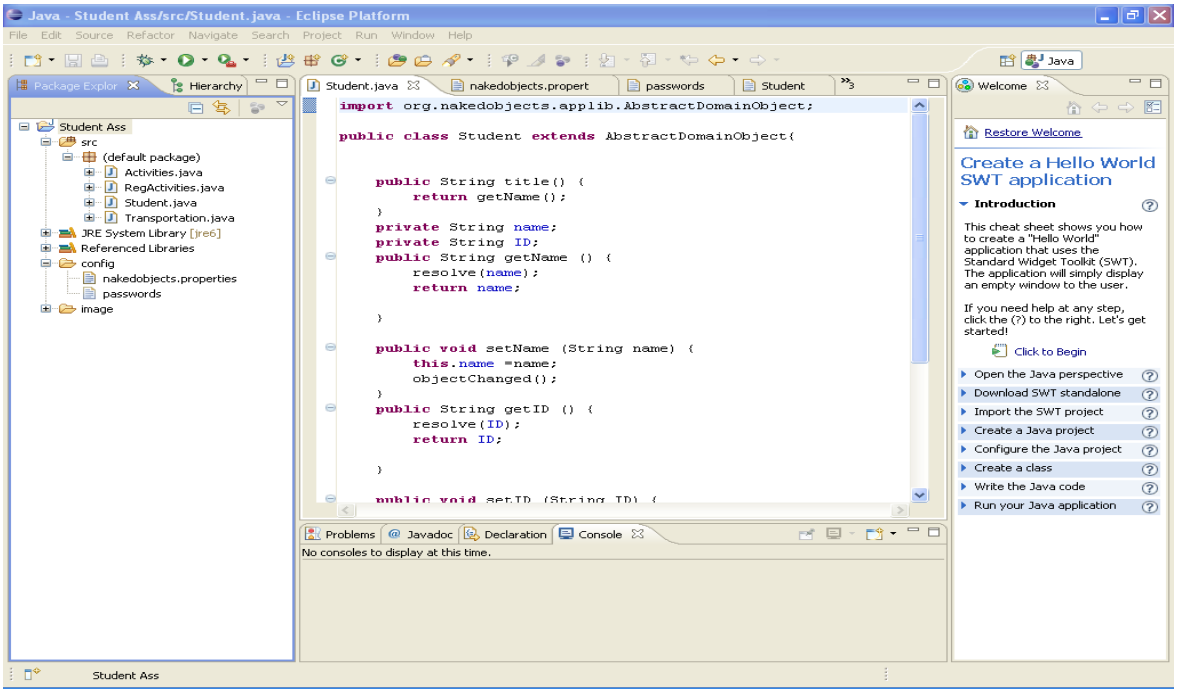


Figure Appendix 5- 3: Java Code through Eclipse Screen Shot

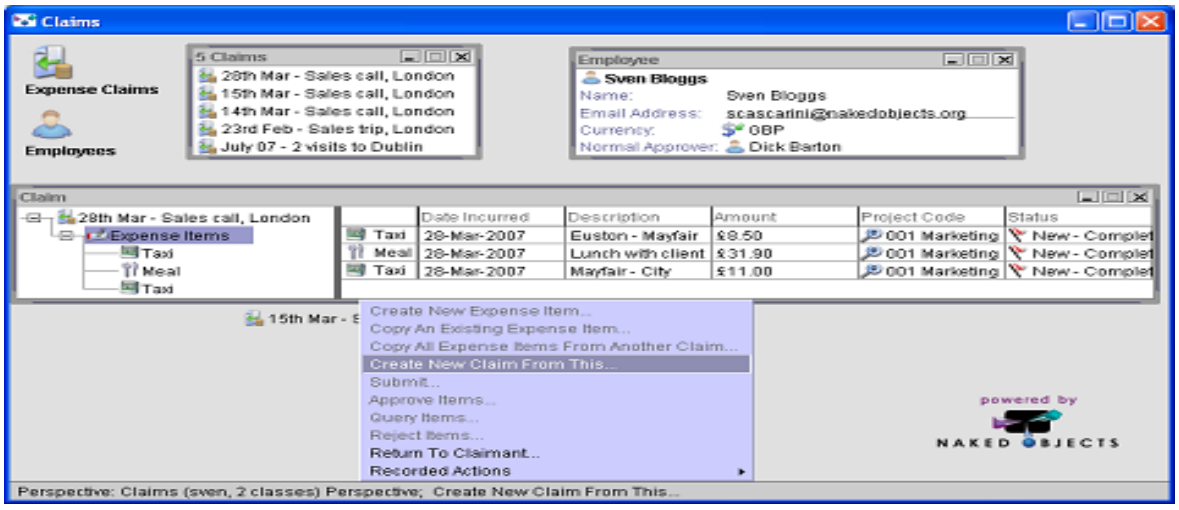


Figure Appendix 5- 4: Drag and Drop Screen Shot

Appendix 6

Use case proforma of SLCS (post-graduation)

Use Case No. 02	Use Case Name: Import Monthly Report
Brief Description: The Recruitment Co-ordinator will import monthly Excel report file through Import Wizard	
Actor: Recruitment Co-ordinator	
Frequency of execution: Monthly	
Scalability: Only one user; Recruitment Co-ordinator	
Criticality: Very. Because any changes in the Excel file format may cause data validation problems in database.	
Primary Path: 1. Recruitment Co-ordinator runs the import wizard 2. A File Open Dialog Box will be displayed 3. Recruitment Co-ordinator selects the required Excel file and clicks on Open Button 4. The imported Excel file in original format will be displayed with all records and the total number of records imported. 5. Recruitment Co-ordinator will clicks on Save button to save in the database 6. The imported file will be save along with new schools and courses found those were not present in schools and courses databases respectively by displaying the information about the number of new schools and courses found and total number of records saved. 7. After saving Recruitment Co-ordinator can exit the import wizard by click on close button.	
Alternative: none	
Exception: An Excel file that is already imported will generate primary key violation error	
Note: Created by Saraj Din, August, 09	

Table Appendix 6-1: Proforma for Use Case Import Monthly Report

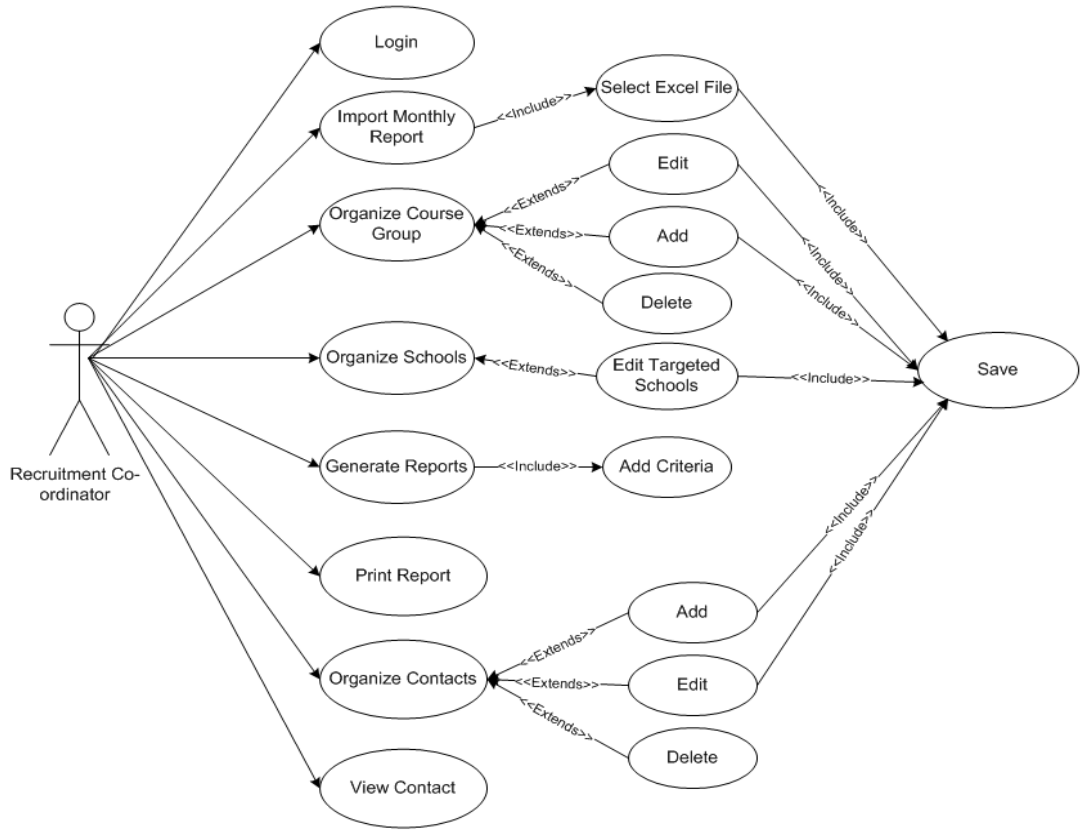


Figure Appendix 6- 2: Use Case Diagram Prepared by Din (2009)

Use Case No. 03	Use Case Name: Organize Course Group
Brief Description: The Recruitment Co-ordinator will organize course groups as well as courses under each course by adding, editing and deleting both courses and groups	
Actor: Recruitment Co-ordinator	
Frequency of execution: Daily or as needed	
Scalability: Any number of course groups and courses under each could be added	
Criticality: Normal. Only deleting a course group or course for which student applications exist may generate foreign key violation error.	
Primary Path: 1. Recruitment Co-ordinator will open Course Group form 2. All course groups will be displayed in a combo box and under each group courses will be displayed belonging to that group in a read only mode. 3. Recruitment Co-ordinator will add a new course group by clicking Add New Group button 4. Recruitment Co-ordinator will edit an existing course by clicking on Edit button 5. Recruitment Co-ordinator will clicks on Delete button to delete a course group 6. A confirmation message will be displayed, if user confirms, and no application under that course group exists then it will be deleted from the database. 7. Under a course group Recruitment Co-ordinator will add, edit or delete a course 8. Recruitment Co-ordinator will cancel add, edit by clicking on Cancel button 9. Recruitment Co-ordinator can exit this form by click on close button.	
Alternative: none	
Exception: not known	
Note: Created by Saraj Din, August, 09	

Table Appendix 7-3: Proforma for Use Case Organize Course Group

Use Case No. 07	Use Case Name: Organize Contacts
Brief Description: The Recruitment Co-ordinator will organize contacts details by adding, editing and deleting both courses and groups	
Actor: Recruitment Co-ordinator	
Frequency of execution: Daily or as needed	
Scalability: Any number of contacts could be added	
Criticality: Not known	
Primary Path: 1. Recruitment Co-ordinator will open Contacts form 2. All contacts will be displayed in a read only mode 3. Recruitment Co-ordinator will add a new contact by clicking Add New Group button 4. Recruitment Co-ordinator will edit an existing contact by clicking on Edit button 5. Recruitment Co-ordinator will clicks on Delete button to delete a contact 6. A confirmation message will be displayed, if user confirms, then it will be deleted from the database. 7. Recruitment Co-ordinator will cancel add, edit by clicking on Cancel button 8. Recruitment Co-ordinator can exit this form by click on close button	
Alternative: none	
Exception: not known	
Note: Created by Saraj Din, August, 09	

Table Appendix 7-4: Proforma for Use Case Organize Contacts

Appendix 7

Use case Proforma for PTS (post-graduate)

Use case ID:	001
Use case Name:	Add / Edit Tutor
Brief description:	Option available to Lecturer to be able to add more students on the Tutor list.
Actor:	Lecturer
Scalability:	Can take four users, between Lecturers and management
Frequency of execution	No limits
Criticality:	Very critical as a mistake in entering a failed student as a tutor affect integrity of the application since sessions can be booked with a student who is not a tutor hence can't carry out session as not trained and might even be struggling with the module.
Primary Path:	1. Lecturer logs in online 2. Enters details of students that have done well previously or in the last year with tutor privileges 3. Enter details of area of expertise 4. Generates an email to advise new tutor that bookings will not start coming through
Secondary Path:	Done on the offline systems and enters details as primary path. Details will be entered directly into the database.
Exception:	A tutor cannot be entered twice as a violation error will appear against the student ID.
Notes:	If student has any special requirements so that arrangements can be made.

Table Appendix 7-1: Proforma for Use Case Add New / Edit Tutor

Use case ID:	002
Use case Name:	Add / Edit Tutee
Brief description:	Option available to Lecturer to be able to add more students on the tutor referral list.
Actor:	Lecturer
Scalability:	Can take four users, between Lecturers and management
Frequency of execution	No limits
Criticality:	Very critical as a mistake in entering a passed student who doesn't require support as a tutee can affect efficiency of the application as sessions can be booked with a student who doesn't need the support hence using up time that would be beneficial to someone who needs it.
Primary Path:	1. Lecturer logs in online 2. Enters details of students that have failed previously 3. Generates an email to advise new tutee that bookings will not start coming through
Secondary Path:	Done on the offline systems and enters details as primary path. Details will be entered directly into the database.
Exception:	A tutee cannot be entered twice as a violation error will appear against the student ID.
Notes:	If student has any special requirements so that arrangements can be made.

Table Appendix 8-7: Proforma for Use Case Add New / Edit Tutee

Use case ID:	003
Use case Name:	Update Diary
Brief description:	Available to Tutors to update their availability to run sessions
Actor:	Tutor
Scalability:	Can up to at least 100 users to start with, so that up to 100 tutors can update their availability simultaneously.
Frequency of execution	No limits
Criticality:	Very critical as when diary updated incorrectly it can lead to not valid booking being made.
Primary Path:	1. Tutor logs in online 2. A calendar will be displayed and there he can load availability 3. Dialogue box comes up to confirm availability
Secondary Path:	none
Exception:	When removed as tutor, facilities will b withdrawn.
Notes:	If student has any special requirements so that arrangements can be made.

Table Appendix 8-7: Proforma for Use Case Update Diary

Use case ID:	004
Use case Name:	Add room
Brief description:	Available to lectures and management load more rooms in the system to make a provision for a booking
Actor:	Tutor
Scalability:	Can take four users. Between Lecturers and management
Frequency of execution	No limits
Criticality:	Very critical as when no rooms are available, a booking can't be done
Primary Path:	1. Lecturer logs in online 2. Loads rooms that are available to make a booking 3. Dialogue box comes up to confirm availability
Secondary Path:	Can use offline system and enter info straight into database.
Exception:	Room cannot be entered as available twice at same time as room ID will have a violation error against it.
Notes:	If room has any provisions for special arrangements for example, disability access

Table Appendix 7-4: Proforma for Add Room

Use case ID:	005
Use case Name:	Schedule a session
Brief description:	Available to Tutors, tutees and lecturers to enable make a booking
Actor:	Tutor, Tutee and Lecturer
Scalability:	Can take at least 200 users so that system can still withstand peak time bookings for example during break time or when tutor and tutee recruitment has just been done.
Frequency of execution	No limits
Criticality:	Very critical as if this part fails then bookings will not be possible hence the system will be a failure.
Primary Path:	1. Log in online 2. Identify tutor/tutee/module 3. If booking a tutor, check availability 4. Identify room 5. Check room availability 6. When both room and tutor are available, tick options and proceed 7. Dialogue box comes up to confirm actions 8. System automatically sends a message to tutor/tutee to advise of booking
Secondary Path:	Lecturer use offline system and book straight into database.
Exception:	User cannot make two bookings for same time as student ID will have a violation error against it.
Notes:	If need any special arrangements to assist attendance

Table Appendix 7-5: Proforma for Schedule Session

Use case ID:	006
Use case Name:	Attendance register
Brief description:	Available to tutees so they can show if the booked sessions are being attended
Actor:	Tutee
Scalability:	Can take at most 100 users to allow flexibility for peak times.
Frequency of execution	Only available after a session booked time has gone past
Criticality:	Very critical as it will help to see success of the system
Primary Path:	1. Log in online 2. Go on past sessions screen 3. Click mark attendance
Secondary Path:	None.
Exception:	Attendance can only be marked once as it will not be available once first marked.
Notes:	If marked not attended, put reasons why not attended.

Table Appendix 7-6: Proforma for Marking an Attendance Register

Use case ID:	007
Use case Name:	Calculate rewards
Brief description:	Available to Lecturers so that they can calculate the rewards a tutor has accumulated for delivering sessions
Actor:	Lecturer
Scalability:	Can take at least four users.
Frequency of execution	Once per month
Criticality:	Critical as it's an incentive for tutors to keep delivering the sessions
Primary Path:	1. Log in online 2. Check register for tutee attendance 3. Calculate reward as originally agreed with tutor 4. Allocate reward to tutor
Secondary Path:	Offline system same as primary path
Exception:	Not known
Notes:	Date this calculation is correct

Table Appendix 7-7: Proforma for Calculate Rewards

Appendix 8

TrueView implementation for PTS (post-graduate)

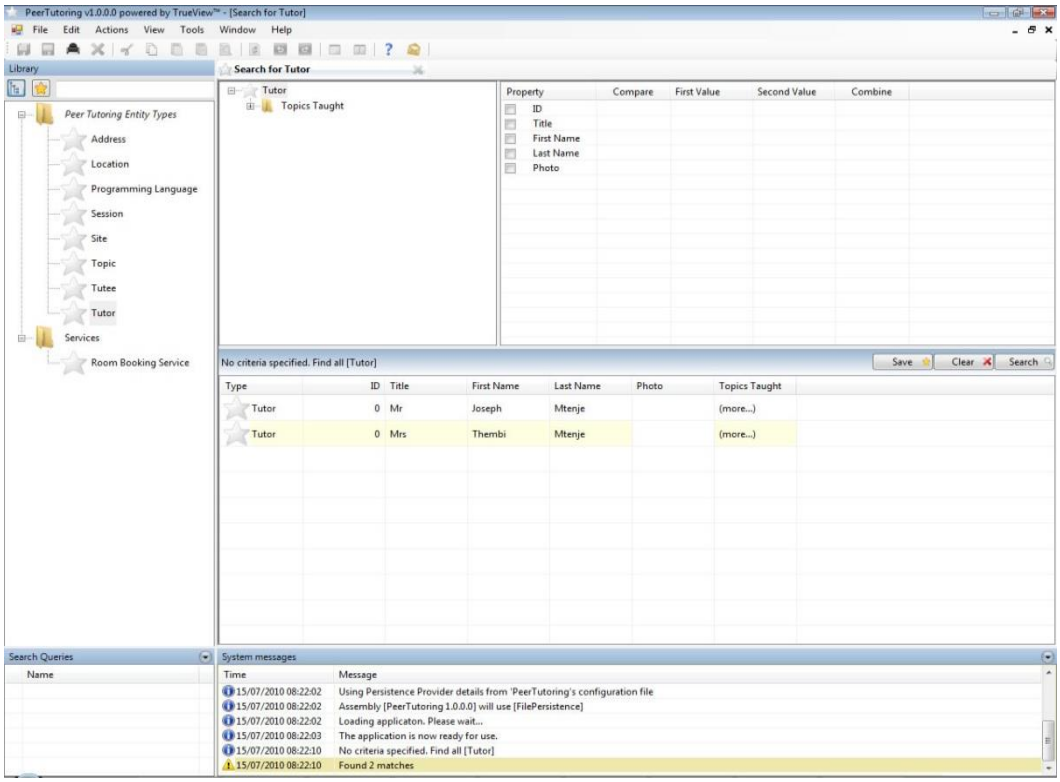


Figure Appendix 8- 1: Screen Shot - Tutor's Availability

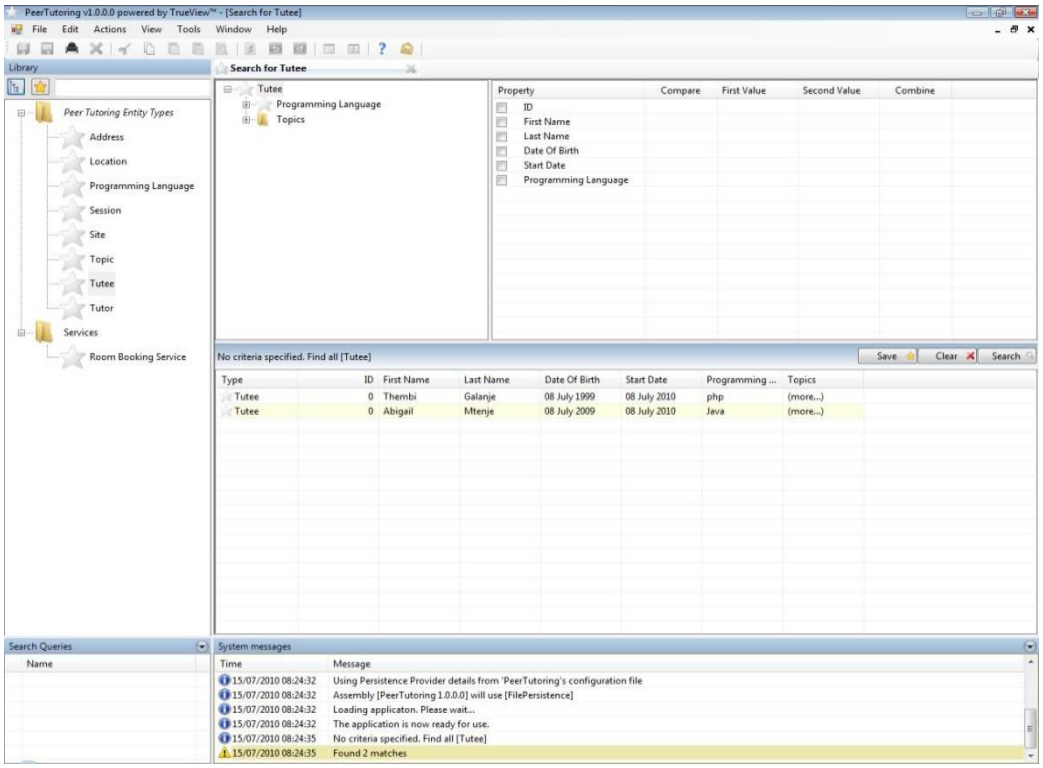


Figure Appendix 8- 2: List of Tutees needing Support in Programming